



Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😊 

» Algoritmos

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 :

- ✓ Estructura de control anidada
- ✓ Estructura de control múltiple

LEARNING PATHWAY

4.2

Start! 🏁

Fundamentos de la
programación en Java

Estructuras de control repetitivas:
Mientras y Repetir

Estructura Mientras

Calculadora en
Bucle

Estructura Repetir

¿Es múltiplo?

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Reconocer la importancia de la utilización de las de estructuras de control repetitivas.



Comprender la implementación de la estructura Mientras.



Entender la implementación de la estructura Repetir.





Rompehielo 🥶

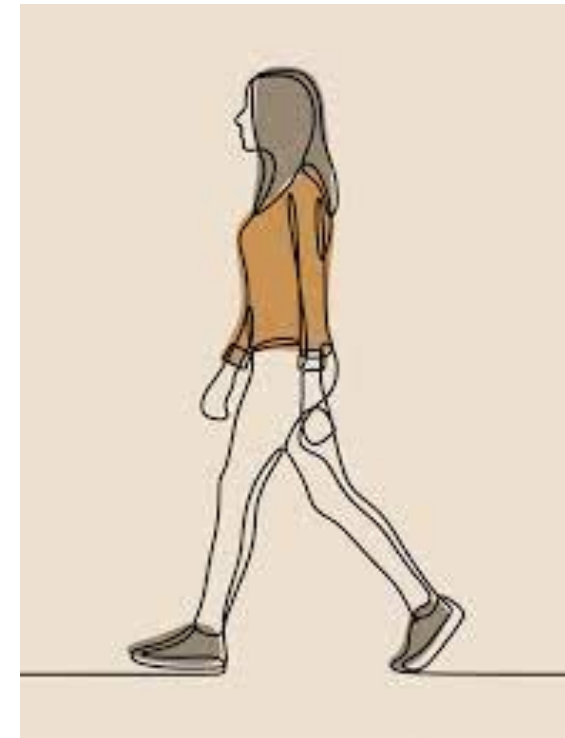


Caminando: 🙌

La persona que vemos en la imagen necesita caminar 3 km.... pero **por cada vez que vemos esta imagen**, solo está caminando 1 km.

Consigna: ✍️ Respondan en el chat

¿Cómo podemos hacer para que **gráficamente** camine 3km?

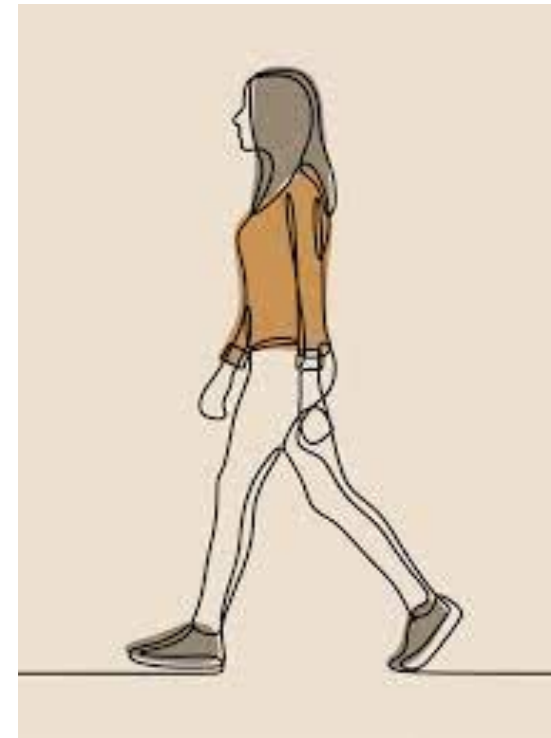
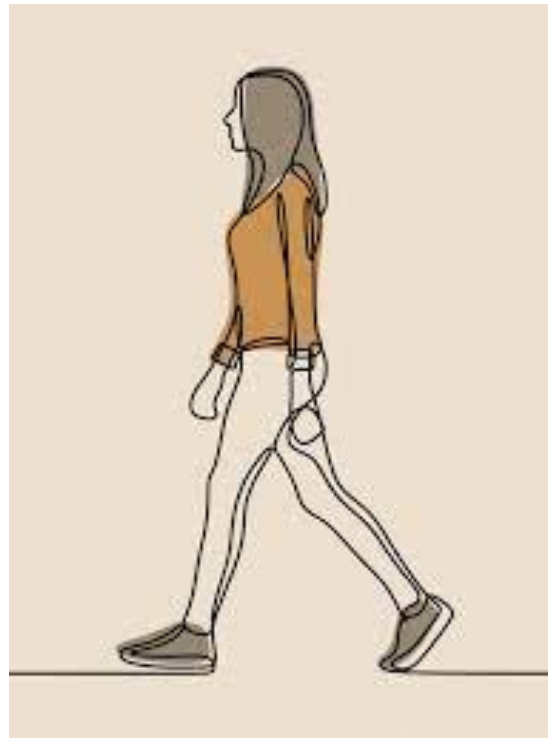
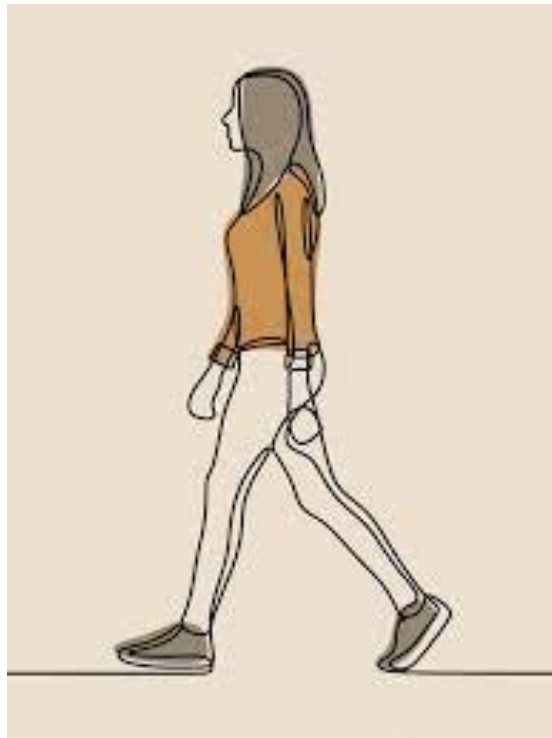




Rompehielo 🥶

¿Tiene lógica que pensemos la siguiente representación!

Respondan en el chat: ¿Cómo harían esto en programación?




› Estructuras de Control Repetitivas



Estructuras Repetitivas

¿Para qué sirven?

Las estructuras repetitivas o bucles son construcciones de programación que **permiten ejecutar un bloque de código varias veces** de forma controlada.

Permiten repetir un proceso sin tener que escribir el código muchas veces. Simplifican la escritura de código y se pueden controlar para que se repitan hasta cumplir cierta condición (**mientras**), un número fijo de veces (**para**) o de forma indefinida (**repetir**). 

Resultan muy útiles para operaciones reiterativas como procesar listas de elementos, lecturas desde archivos, interacciones con el usuario, etc. Mediante los bucles se puede recorrer arrays, strings, objetos, estructuras de datos complejas, etc.

Hay que tener cuidado de no crear bucles infinitos y controlar adecuadamente la condición de salida.

› Estructura Mientras



Bucle Mientras

¿Para qué sirve?:

El bucle **Mientras** (while) permite **repetir la ejecución de un bloque de código mientras se cumpla una condición**. La condición se evalúa **antes de ingresar al bucle**. El funcionamiento es el siguiente:

- Primero se evalúa la condición dentro del paréntesis.
- Si la condición se cumple (es **verdadera**), se ejecuta el código definido.
- Una vez termina la ejecución del bloque de código, se vuelve a evaluar la condición.
- Si sigue siendo verdadera, se repite la ejecución del código nuevamente.
- Este proceso se repite iterativamente mientras la condición se cumpla.
- Cuando la condición llega a **false**, **se sale del bucle** y continúa la ejecución normal después de éste.

Es importante que la condición en algún momento se haga falsa para evitar un bucle infinito.



Bucle Mientras

Mientras:

Se ejecuta siempre que una **condición sea verdadera**.
Imagina un reloj despertador que suena hasta que presionas el botón de apagado. Mientras la condición de que el reloj siga sonando sea verdadera, el bucle se repetirá.

Mientras que(**condición**)

acción 1

acción 2

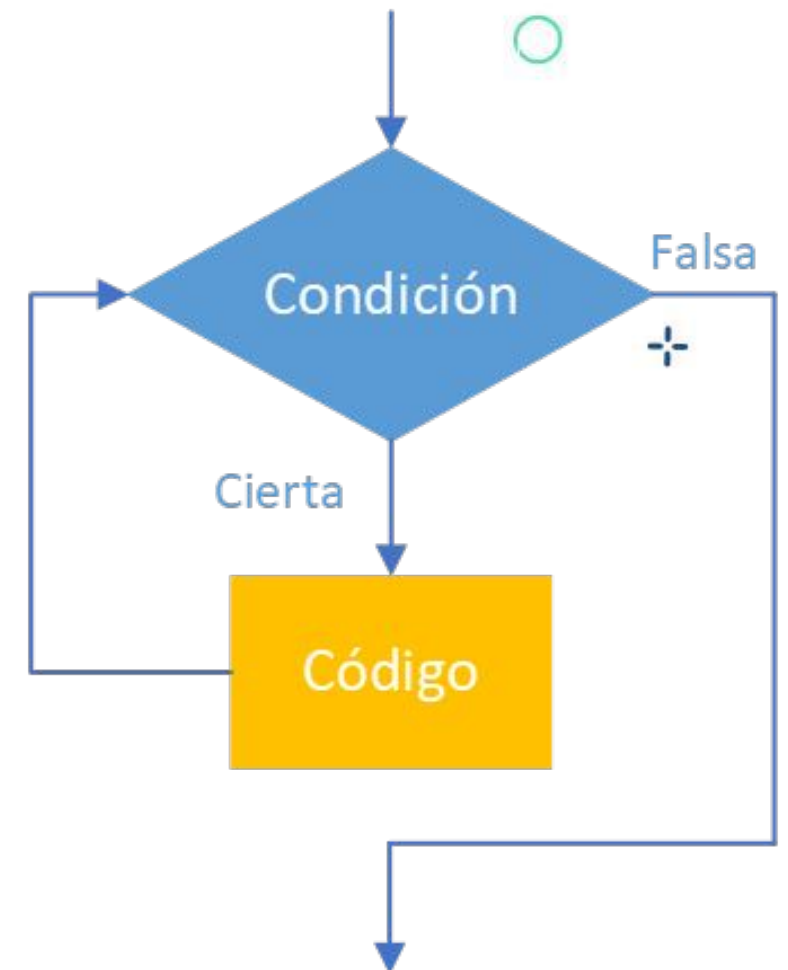
acción 3

... acción n

Fin Mientras



Bucle condicional



LIVE CODING

Ejemplo en vivo

*Vamos a escribir un algoritmo 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**.*

- 1. Variables: límite, num, suma.*
- 2. Pedir el límite, y luego colocarlo como condición del bucle Mientras*
- 3. Sumar cada número ingresado (reemplazando el valor de “num”) hasta que supere el valor ingresado como límite.*

 **Tiempo: 20 minutos**





Ejercicio N° 1

Calculadora en bucle



Calculadora en bucle

Mejorando la calculadora: 🙌

La clase anterior creamos una calculadora con estructura múltiple. En este ejercicio, vamos a mejorar ésta herramienta utilizando el bucle Mientras.

Consigna: 🖋 Realizar un algoritmo que pida dos números enteros positivos por teclado y muestre por pantalla el siguiente menú de opciones de operaciones que se pueden realizar con los números ingresados. El usuario deberá elegir una opción y el programa deberá mostrar el resultado por pantalla, **y luego volver a mostrar el menú de opciones para elegir otra operación**. Si la opción elegida es la número 5 (salir), **se debe detener la ejecución**.

1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. **Salir**

Tiempo 🕒: 20 minutos



Calculadora en bucle



Paso a paso:

1. Declarar las variables num1, num2 y opción.
2. Pedir al usuario dos números enteros (num1 y num2).
3. Pedir al usuario la opción que desea del menú (cada opción representa una operación aritmética).
4. Dentro de un bucle Mientras: usar una estructura Según para evaluar la opción.
5. Mostrar el resultado de la opción elegida.
6. Si la opción elegida es distinta de 5, volver a mostrar el menú y seguir realizando operaciones. De lo contrario, detener la ejecución.



› Estructura Repetir



Bucle Repetir

¿Para qué sirve?:

El bucle **Repetir** (do while) permite ejecutar un bloque de código repetidamente mientras se cumpla una condición **al final del bucle**. El funcionamiento es:

- Primero se ejecuta el código dentro de las llaves {} una vez.
- Al final, se evalúa la condición dentro del while.
- Si la condición se cumple (es verdadera), se vuelve a ejecutar el código del bucle desde el principio.
- El proceso se repite mientras la condición siga siendo verdadera.
- Cuando la condición se vuelve falsa, se sale del bucle y continúa la ejecución normal.

Al evaluar la condición al final, el código se ejecuta al menos una vez siempre.
Hay que evitar condiciones siempre verdaderas para no causar bucles infinitos.



Bucle Repetir

Repetir:

Es similar al "Mientras", **pero siempre se ejecuta al menos una vez** antes de verificar la condición. Es como una montaña rusa que siempre hace un recorrido completo antes de verificar si debe continuar.

Repetir

acción 1

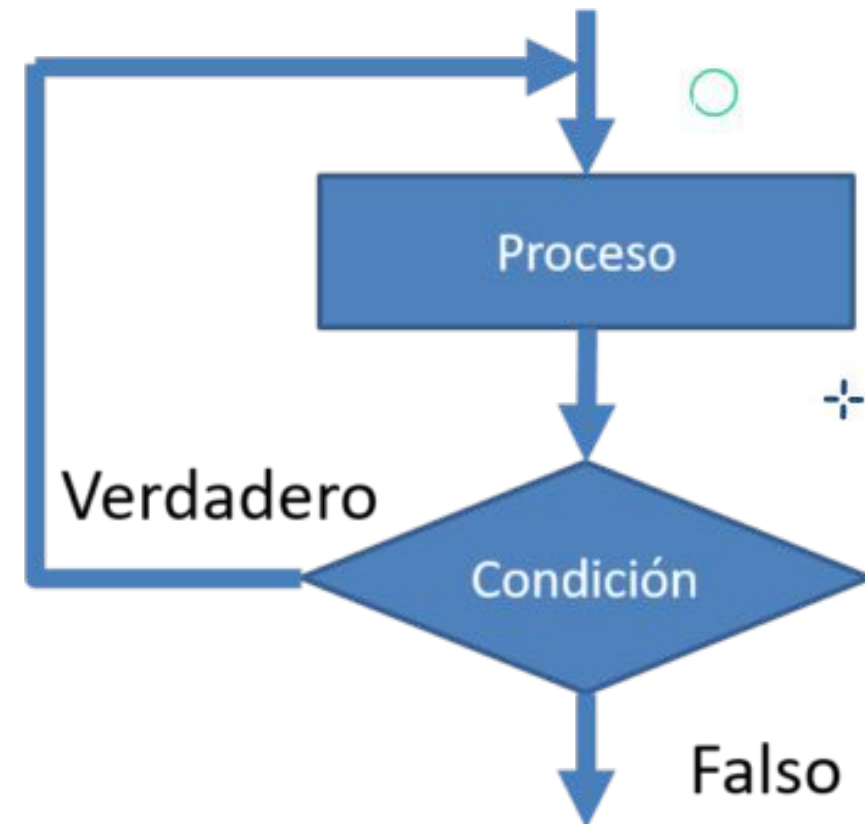
acción 2

acción 3

... acción n

Mientras que(**condición**)

Fin_Repetir



LIVE CODING

Ejemplo en vivo

*Vamos a realizar un programa que calcule y visualice el valor **máximo**, el valor **mínimo** y el **promedio** de n números ($n > 0$). El valor de n se solicitará al principio del programa (**cantidad**) y los números serán introducidos por el usuario.*

- 1. Crear variables: num, cantidad, minimo, máximo, promedio.*
- 2. Crear un bucle Repetir con la condición evaluando la cantidad.*
- 3. Mostrar por pantalla el valor del máximo, minimo y promedio.*

 **Tiempo:** 20 minutos





Ejercicio N° 2

¿Es múltiplo?



¿Es Múltiplo?

La condición va al final! 🙌

El bucle repetir nos sirve para ahorrar código y realizar iteraciones. En este ejercicio vamos a implementar este ciclo de repetición.

Consigna: 📝

Escribir un programa que lea números enteros. Si el número es múltiplo de cinco debe detener la lectura y mostrar la cantidad de números leídos (contador), la cantidad de números pares y la cantidad de números impares.

Tiempo 🕒: 25 minutos

Paso a paso: ⚙️

- Crear variables: num, contador, pares, impares.
- La condición será evaluada según una operación matemática. Si el número es múltiplo de 5, debe salir del bucle e imprimir los datos solicitados.

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ Reconocer las estructuras de control repetitivas.
- ✓ Implementar el ciclo de repetición Mientras.
- ✓ Implementar el ciclo de repetición Repetir.
- ✓ Comprender cuándo utilizar cada bucle.

#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)
 - b. Lectura Módulo 4, Lección 2: página 16
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.

