Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😁

alkemy

Clase grabada

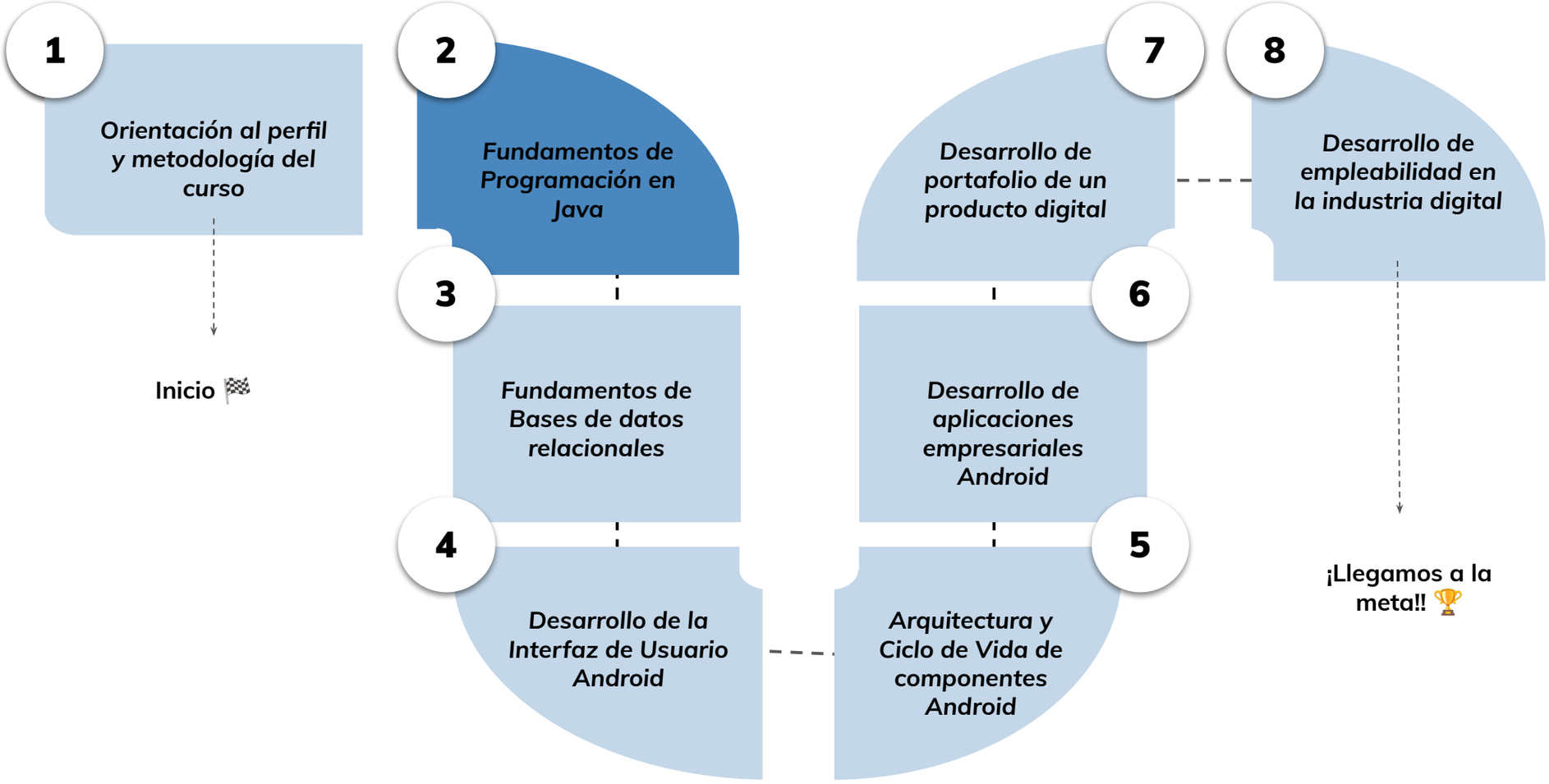
El entorno java para la programación

AE3.1: Utilizar la sintaxis básica del lenguaje java para la construcción de programas que resuelven un problema de baja complejidad.

alkemy

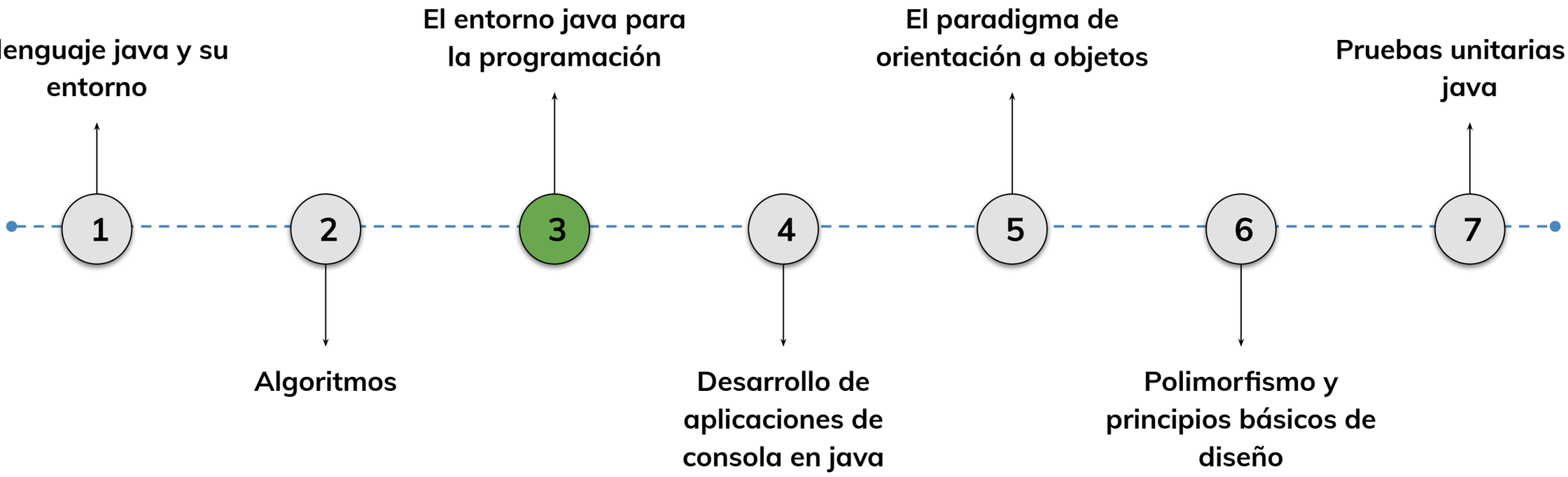
Hoja de ruta

*¿Cuáles **módulos** conforman el programa?*



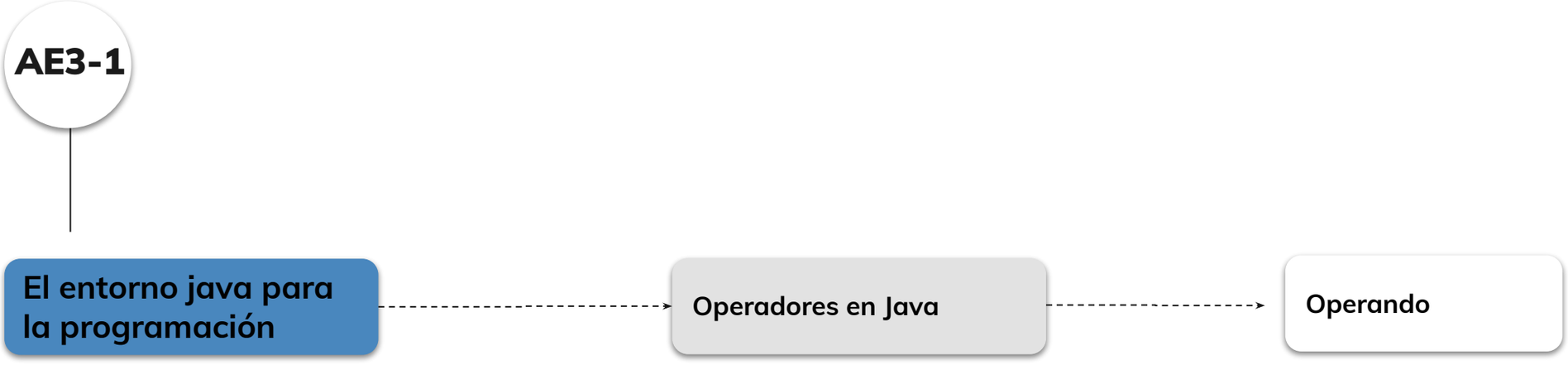
Roadmap de lecciones

*¿Cuáles **lecciones** estaremos estudiando en este módulo?*



Learning Path

¿Cuáles temas trabajaremos hoy?



Operadores en Java:
Aritméticos
Unarios
Relacionales
Condicionales

alkemy

Objetivos de aprendizaje

¿Qué aprenderás?

- Reconocer la implementación y manipulación de los operadores en Java

< > Rompehielo🥶

Compartamos experiencias: 🙌

Levanten la mano para ser voluntarios de la calculadora humana! La idea es que vayas escribiendo todas las operaciones que pondremos como ejemplo. Serás la pantalla de la calculadora. Las operaciones serán aritméticas y lógicas.

Consigna:

El moderador dirá en voz alta una operación matemática simple, por ejemplo: "2 + 4"

Los participantes deberán indicar en el chat qué operador aritmético corresponde: en este caso, el signo "+".

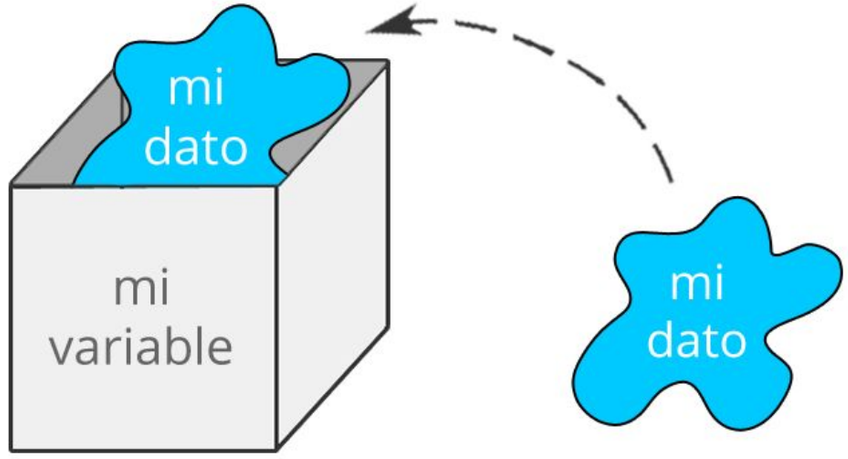
El voluntario anotará en su "pantalla" la operación con el operador que los participantes indiquen: 2 + 4

Los participantes deberán identificar el operador correcto para que el voluntario lo añada.

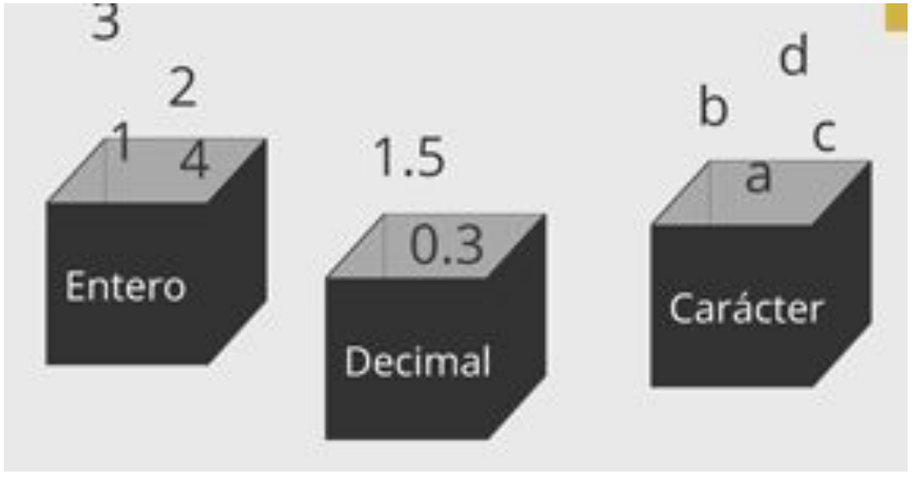
Variables y tipos de Datos

¿Qué es una variable?

- En programación, una variable es un espacio reservado en memoria para almacenar un dato.
- Debe tener un nombre (o etiqueta) dado por el usuario, y un tipo específico.
- **Ejemplo**
  - Ciudad = “Viña del Mar”



Variables y tipos de dato

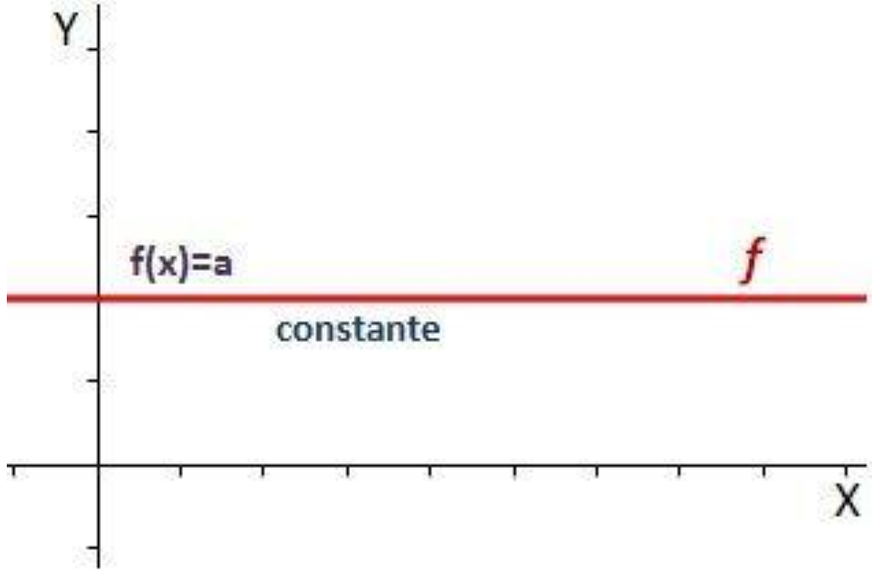


- Es una restricción para ciertos tipos de variables, que permite: manipulación, interpretación y representación.

- Los más comunes son:
  - **Numéricos:** enteros o decimales.
  - **Caracteres:** unidad mínima de escritura.
  - **Alfanumérico:** Caracteres del código ASCII (letras, números y símbolos). Pueden ser manipulados como letra o como número.
  - **Booleano:** valores binarios, representados generalmente como Verdadero o Falso.

¿Qué es una constante?

- Usa un espacio en memoria, al igual que las variables.
- El dato almacenado no se puede cambiar.
- Puede venir definido en el lenguaje de programación, o establecido por el programador.
- **Ejemplo**
  - Pi = 3.1415926535 (π)



Operadores en Java

Operadores en Java

¿Qué son los operadores?:

En programación un operador representa un símbolo que permite realizar operaciones aritméticas, relacionar elementos o hacer preguntas donde se involucra más de una condición.

Un **operador lleva a cabo operaciones** sobre uno (operador unario), dos (operador binario) o tres (operador ternario) datos u operandos de tipo primitivo devolviendo un valor determinado también de un tipo primitivo. El tipo de valor devuelto tras la evaluación depende del operador y del tipo de los operandos. Por ejemplo, los operadores aritméticos trabajan con operandos numéricos, llevan a cabo operaciones aritméticas básicas y devuelven el valor numérico correspondiente.

Operadores en Java

Operadores aritméticos

Los operadores aritméticos en Java son aquellos que nos permiten realizar operaciones matemáticas: **suma**, **resta**, **multiplicación**, **división,resto.**

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	operador unario de cambio de signo	`-4`	`-4`
+	Suma	`2.5 + 7.1`	`9.6`
-	Resta	`235.6 - 103.5`	`132.1`
*	Producto	`1.2 * 1.1`	`1.32`
/	División (tanto entera como real)	`0.050 / 0.2` `7 / 2`	`0.25` `3`
%	Resto de la división entera	`20 % 7`	`6`

Operadores en Java

Operadores unarios

Los operadores unarios en Java son aquellos que solo requieren un operando para funcionar.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

++	Incremento
	`i++` primero se utiliza la variable y luego se incrementa su valor
	`++i` primero se incrementa el valor de la variable y luego se utiliza
		`4++`
		`a=5;`
		`b=a++;`
		`a=5;`
		`b=++a;`
			`5`
			
			`a vale 6 y b vale 5`
			
			`a vale 6 y b vale 6`

--	decremento	`4--`	`3`

Operadores en Java

Operadores de relación

Realizan comparaciones entre datos compatibles de tipos primitivos (numéricos, carácter y booleanos) teniendo siempre un resultado booleano. Los operandos booleanos sólo pueden emplear los operadores de igualdad y desigualdad.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	igual que	`7 == 38`	`false`
!=	distinto que	`'a' != 'k'`	`true`
<	menor que	`'G' < 'B'`	`false`
>	mayor que	`'b' > 'a'`	`true`
<=	menor o igual que	`7.5 <= 7.38`	`false`
>=	mayor o igual que	`38 >= 7`	`true`

Operadores en Java

Operadores lógicos

Realizan operaciones sobre datos booleanos y tienen como resultado un valor booleano.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	`!false` `!(5==5)`	`true` `false`
|	Suma lógica – OR (binario)	`true | false` `(5==5)|(5<4)`	`true` `true`
^	Suma lógica exclusiva – XOR (binario)	`true ^ false` `(5==5)|(5<4)`	`true` `true`
&	Producto lógico – AND (binario)	`true & false` `(5==5)&(5<4)`	`false` `false`
||	Suma lógica con cortocircuito: si el primer operando es `true` entonces el segundo se salta y el resultado es `true`	`true || false` `(5==5)||(5<4)`	`true` `true`
&&	Producto lógico con cortocircuito: si el primer operando es `false` entonces el segundo se salta y el resultado es `false`	`false && true` `(5==5)&&(5<4)`	`false` `false`

Operadores en Java

Operadores lógicos

Para mejorar el rendimiento de ejecución del código es recomendable emplear en las expresiones booleanas el operador SS en lugar del operador s. En este caso es conveniente situar la condición más propensa a ser falsa en el término de la izquierda.

Esta técnica puede reducir el tiempo de ejecución del programa.

De forma equivalente es preferible emplear el operador || al operador |. En este caso es conveniente colocar la condición más propensa a ser verdadera en el término de la izquierda.

LIVE CODING

Practicando operaciones:

Vamos a realizar diversas operaciones en Java en su IDE, y vamos a imprimir por pantalla los resultados de dichas operaciones:

1. *Desarrollar operaciones aritméticas simples (suma, resta, multiplicación y división).*
2. *Desarrollar operaciones de incremento, utilizando el operador ++.*
3. *Desarrollar operaciones lógicas, utilizando && y ||*

Tiempo: 25 Minutos

LIVE CODING

```
public class Operaciones {

    public static void main(String[] args) {
        // Definir variables
        int num1 = 10;
        int num2 = 5;
        int resultado;

        // 1. Operaciones Aritméticas Simples
        // Suma
        resultado = num1 + num2;
        System.out.println("Suma: " + num1 + " + " + num2 + " = " + resultado);

        // Resta
        resultado = num1 - num2;
        System.out.println("Resta: " + num1 + " - " + num2 + " = " + resultado);

        // Multiplicación
        resultado = num1 * num2;
        System.out.println("Multiplicación: " + num1 + " * " + num2 + " = " + resultado);
```

LIVE CODING

```
        // División
        if (num2 != 0) { // Verificar que no se divida por cero
            resultado = num1 / num2;
            System.out.println("División: " + num1 + " / " + num2 + " = " + resultado);
        } else {
            System.out.println("Error: División por cero");
        }

        // 2. Operaciones de Incremento (Operador ++)
        System.out.println("\nOperaciones de Incremento:");
        System.out.println("Valor inicial de num1: " + num1);
        num1++; // Incrementar en 1
        System.out.println("Después de incrementar num1 (num1++): " + num1);

        // 3. Operaciones Lógicas (&& y ||)
        System.out.println("\nOperaciones Lógicas:");

        // Operador lógico AND (&&)
        boolean andResult = (num1 > 5) && (num2 < 10);
        System.out.println("Resultado de (num1 > 5) && (num2 < 10): " + andResult);

        // Operador lógico OR (||)
        boolean orResult = (num1 > 15) || (num2 < 10);
        System.out.println("Resultado de (num1 > 15) || (num2 < 10): " + orResult);
    }
}
```

Ejercicio N° 1

Operando

Operando

Contexto: 🙌

En la programación, la necesidad de realizar operaciones es constante. Es por esto que te invitamos a practicar los conceptos vistos en tu IDE Eclipse. Debes mostrar todos los resultados por pantalla.

Consigna: ✍️

- Pedir al usuario dos números y mostrar su suma
- Leer un número y mostrar su cuadrado elevándolo al exponente 2
- Leer dos palabras y concatenarlas en una frase usando el operador +
- Pedir dos números y comparar si son iguales con == mostrando el resultado

Tiempo🕐: *20 minutos*

Momento:

Time-out!

⏰5 -10 min.

alkemy

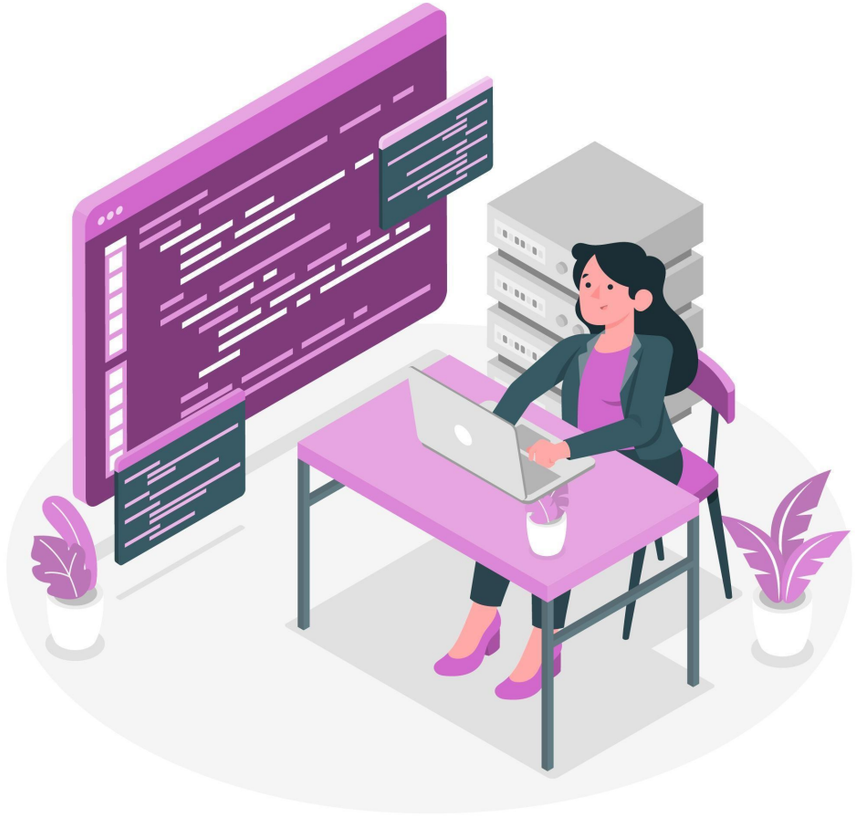
Expresiones

Expresiones

¿Qué es una expresión?

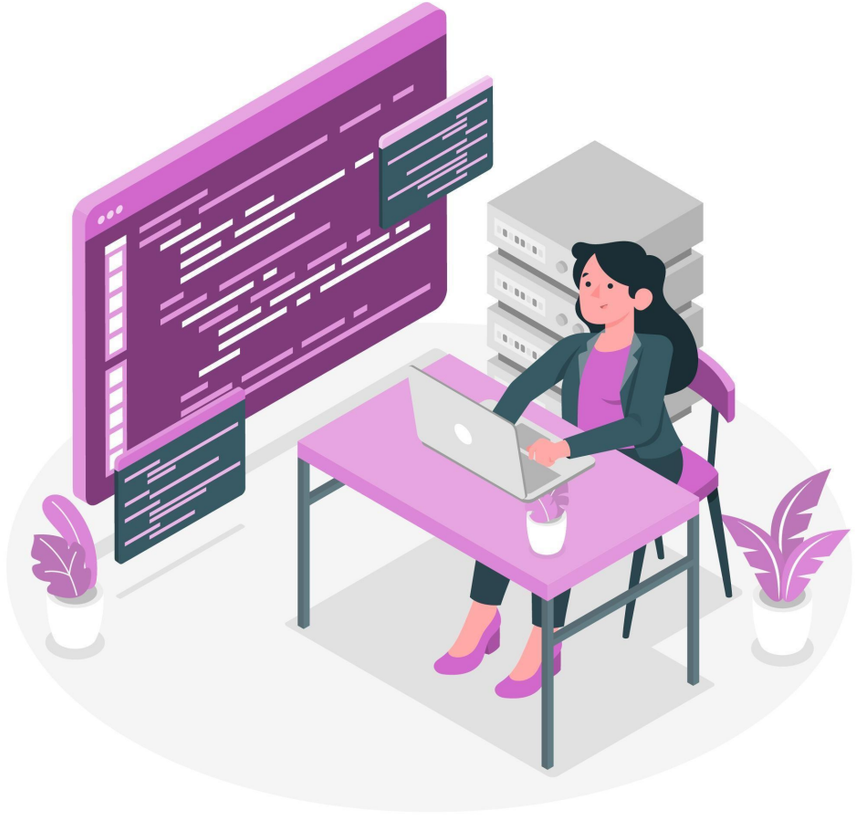
Una expresión es un conjunto de variables, operadores e invocaciones de métodos que se construyen para poder ser evaluadas retornando un resultado.

Cuando tengamos expresiones de evaluación complejas es recomendable que utilicemos paréntesis para saber cual es el orden de ejecución de operaciones. En el caso de no utilizar paréntesis se ejecutará el orden de preferencia de operadores. Por ejemplo, la división tiene más preferencia que la suma.



Expresiones

Si dos operadores se encuentran en la misma expresión, el orden en el que se evalúan puede determinar el valor de la expresión. En la siguiente tabla se muestra el **orden o prioridad en el que se ejecutan los operadores** que se encuentren en la misma sentencia. Los operadores de la misma prioridad se evalúan de izquierda a derecha dentro de la expresión.



Expresiones

Prior.	Operador	Tipo de operador	Operación
1	++ -- +, - ~ ¡	Aritmético Aritmético Aritmético Integral Booleano	Incremento previo o posterior (unario) Incremento previo o posterior (unario) Suma unaria, Resta unaria Cambio de bits (unario) Negación (unario)
2	(tipo)	Cualquiera	
3	*, /, %	Aritmético	Multiplicación, división, resto
4	+, - +	Aritmético Cadena	Suma, resta Concatenación de cadenas
5	<< >> >>>	Integral Integral Integral	Desplazamiento de bits a izquierda Desplazamiento de bits a derecha con inclusión de signo Desplazamiento de bits a derecha con inclusión de cero
6	<, <= >, >= instanceof	Aritmético Aritmético Objeto, tipo	Menor que, Menor o igual que Mayor que, Mayor o igual que Comparación de tipos
7	== ¡= == ¡=	Primitivo Primitivo Objeto Objeto	Igual (valores idénticos) Desigual (valores diferentes) Igual (referencia al mismo objeto) Desigual (referencia a distintos objetos)
8	& &	Integral Booleano	Cambio de bits AND Producto booleano
9	^ ^	Integral Booleano	Cambio de bits XOR Suma exclusiva booleana
10	| |	Integral Booleano	Cambio de bits OR Suma booleana
11	&&	Booleano	AND condicional
12	||	Booleano	OR condicional
13	? :	Booleano, cualquiera, cualquiera	Operador condicional (ternario)
14	= *=, /=, %= +=, -= <<=, >>= >>>= &=, ^=, |=	Variable, cualquiera	Asignación Asignación con operación

Sentencias y Bloques en Java

Sentencias

¿Qué es una sentencia?:

Una sentencia es la **unidad mínima de ejecución de un programa**. Un programa se compone de uno o varios conjuntos de sentencias que acaban resolviendo un problema. En Java, **al final de cada una de las sentencias encontraremos un punto y coma (;).**

- Sentencias de **declaración**:int valor = 2;
- Sentencias de **asignación**: valor = 2;
- Sentencias de **incremento** o **decremento**: valor++;
- Invocaciones a **métodos**: System.out.println("Hola Mundo");
- Creaciones de **objetos**: Circulo miCirculo = new Circulo();
- Sentencias de **control de fiujo**: if (valor>1) { ... };

Bloques

¿Qué es un bloque en Java?:

Cuando hablamos de bloque, en programación, nos referimos a un **trozo de código que está delimitado** de alguna forma. **En Java, son unas llaves { }**. Una de apertura al principio y otra de cierre al final. Así, indicamos donde empieza ese bloque de código y dónde termina.

Los bloques son conjuntos de sentencias delimitados.

Bloques

Analicemos el código:

Las clases pueden contener entre sus llaves, muchas cosas, entre ellas, métodos, que es lo que empieza en la línea 2. Este método llamado main, es el encargado de ejecutar las instrucciones que hay en las líneas 3, 4, y 5.

El método, al igual que pasa con la clase, se abre en la línea donde se declara (línea 2) con una llave de apertura y se cierra cuando acaban las instrucciones que lleva escritas con la llave de cierre (línea 6).

```
1 public class Main {
2     public static void main(String[] args) {
3         int numero1 = 7;
4         int resultado = 10 + numero1;
5         System.out.print(resultado);
6     }
7 }
```

El método es un bloque de código que está dentro de la clase, por este motivo, está envuelto entre las primeras llaves.

LIVE CODING

Practicando:

Vamos a practicar un código simple para poner a prueba la jerarquía de las operaciones. Levantando la mano, los invitamos a contestar

¿Cuál operación se ejecuta primero? ¿Pueden dar otro ejemplo?

1. *int resultado = 2 + 4 * 5;*
2. *int resultado = 2 + (4 * 5);*
3. *int resultado = 5 * 4 / 3;*
4. *int resultado = 5 * (4 + 3);*

Tiempo: 15 Minutos

alkemy

LIVE CODING

```
// Expresión 1: int resultado = 2 + 4 * 5;
Inicio
    resultado1 <- 2 + (4 * 5)  // La multiplicación se realiza primero
    Escribir "Resultado de 2 + 4 * 5: " + resultado1
Fin

// Expresión 2: int resultado = 2 + (4 * 5);
Inicio
    resultado2 <- 2 + (4 * 5)  // El paréntesis obliga a hacer la multiplicación primero
    Escribir "Resultado de 2 + (4 * 5): " + resultado2
Fin

// Expresión 3: int resultado = 5 * 4 / 3;
Inicio
    resultado3 <- (5 * 4) / 3  // Multiplicación primero, luego división
    Escribir "Resultado de 5 * 4 / 3: " + resultado3
Fin

// Expresión 4: int resultado = 5 * (4 + 3);
Inicio
    resultado4 <- 5 * (4 + 3)  // El paréntesis obliga a hacer la suma primero
    Escribir "Resultado de 5 * (4 + 3): " + resultado4
Fin

// Ejemplo adicional
Inicio
    resultado5 <- 6 + 3 * 2  // La multiplicación se realiza antes que la suma
    Escribir "Resultado de 6 + 3 * 2: " + resultado5
Fin
```

Ejercicio N° 2

Corrigiendo

Corrigiendo

Contexto: 🙌

Vamos a ver algunos bloques y sentencias de código a los cuales les faltan algunos delimitadores y operadores. De esta manera, vamos a practicar lo aprendido en la lección.

Consigna: ✍️

- Coloca los signos, símbolos y operadores que faltan en el código.

Tiempo🕐: *20 minutos*

Corrigiendo 1

¡A corregir: 🙌 Completa el código: ¿Qué falta?

```
int edad = 15
System.out.println(edad)
```

Tiempo🕐: *20 minutos*

alkemy

Corrigiendo 2

¡A corregir: 🙌 Completa el código: ¿Qué falta?

```
String nombre "Juan"

System.out.println nombre;
```

Tiempo🕐: *20 minutos*

alkemy

Corrigiendo 3

¡A corregir: 🙌 Completa el código: ¿Qué falta?

```
  public static void main(String[] args) {
int edad 25;


System.out.println edad;
```

Tiempo🕐: *20 minutos*

alkemy

Corrigiendo 4

¡A corregir: 🙌 Completa el código: ¿Qué falta?

```
1 public class Main
2     public static void main(String[] args)
3         byte numero1 = 10
4         byte numero2 = 20
5         boolean comparacion = numero1 != numero2
6         System.out.print(comparacion)
```

Tiempo🕐: *20 minutos*

alkemy

Corrigiendo

Ahora sí, a corregir: 🙌

Ahora sí, a corregir:

Vamos a compartir las respuestas y tambíen a abrir debate: ¿Cuál bloque fue más difícil de corregir ? ¿Dónde tuvieron mayor facilidad? ¿Hubo alguno que no pudieron resolver?

Respuestas: ✍️

1. Punto y coma en ambas líneas
2. Línea 1: operador =, punto y coma. Línea 2: () en nombre
3. Línea 1: operador = . Línea 2: paréntensis. Línea 3: cierre de llaves.
4. Línea 1: llaves de apertura. Línea 1: llaves de apertura. Línea 3, 4, 5 y 6: punto y coma. Línea 7: cierre de llaves.

Tiempo: 🕐 *minutos*

Resumen

¿Qué logramos en esta clase?

- ✓ ***Reconocer la estructura de las expresiones.***
- ✓ ***Comprender la estructura de las sentencias y bloques.***
- ✓ ***Conocer e implementar el uso de operadores en programas Java.***

alkemy

¿Alguna consulta?

alkemy

¡Ponte a prueba!

Momento de ejercitación

Te invitamos a aprovechar esta última sección del espacio sincrónico para realizar de manera individual las ***actividades disponibles en la plataforma.*** *Estas propuestas son clave para afianzar lo trabajado y* ***forman parte obligatoria del recorrido de aprendizaje.***

💪 Análisis de caso ———— 💪 Selección Múltiple

💪 Comprensión lectora

Si al resolverlas surge alguna duda, compartela o tráela al próximo encuentro sincrónico.

alkemy

< ¡Muchas gracias! >