Flujo

Sesión Conceptual 2

{desafío} latam_

- Objetivo de la sesión
- Activación de conceptos clave

- Conceptualización
- Ejercitación

- Reflexión

Inicio

{desafío}
latam_

Objetivo

- Aprender a resolver algoritmos donde se tenga una o dos opciones a resolver
- Crear algoritmos que tomen decisiones en base al valor de una variable
- Conocer operadores de comparación
- Diferenciar asignación de comparación

{desafío} latam_

Desarrollo

{desafío}
latam_

Manejo básico de flujo

La instrucción IF

```
if(condicion){
      //código que se ejecutará solo si se ejecuta el código
}
```

Ejemplo de if

```
System.out.println("¿Qué edad tienes?");
int edad = sc.nextInt();
if(edad >= 18) {
      System.out.println("Eres mayor de edad");
}
```

Operadores de comparación

El resultado puede ser:

- true
- false

```
int a = sc.nextInt();  //1
int b = sc.nextInt();  //6
System.out.println(a > b);  //false
```

Operadores de comparación en números

Operador	Nombre	Ejemplo	Resultado
==	igual a	2 == 2	true
!=	distinto a	2 != 2	false
>	mayor a	3 < 4	false
>=	mayor o igual a	3 >= 3	true
<	menor a	4 < 3	false
<=	menor o igual a	3 <= 4	true

Operadores de comparación en Strings

```
String a = "texto 1";
String b = "texto 2";
System.out.println(a == b);  //false
```

¿Puede ser un texto mayor que otro?

No. En objetos no podemos usar el operador < o >. Al tratar de compilar, obtendremos este error

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
	The operator < is undefined for the argument type(s) java.lang.String,
java.lang.String

	at
MiPrimerPrograma/miprimerprograma.MiPrimerPrograma.main(MiPrimerPrograma.java:11)
```

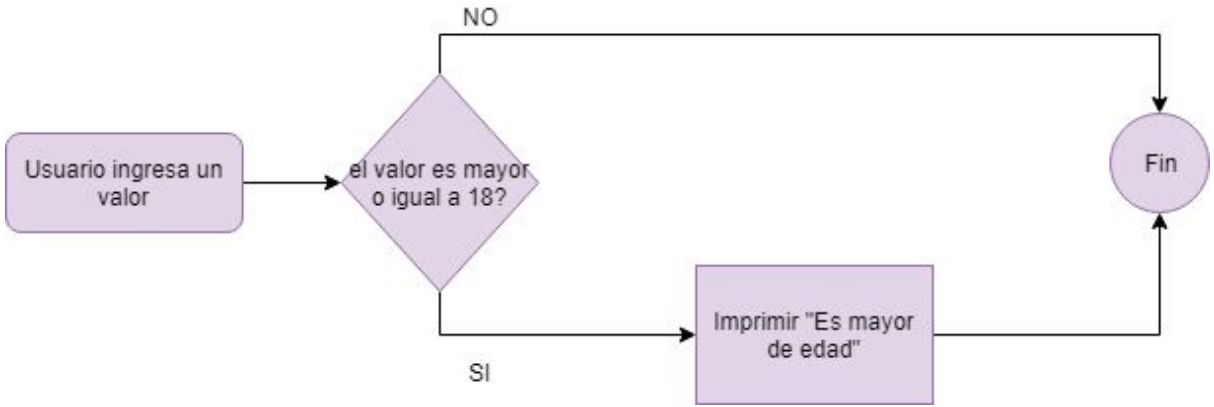
```
String uno = "Hola";
String dos = "Chao";
if(uno.compareTo(dos) < 0) {
	System.out.println("uno es menor");
}
```

compareTo retorna

- 0 si los Strings son iguales
- -1 si el primero es menor,
- 1 si es mayor.

{desafío} latam_

Retomando ejemplo anterior

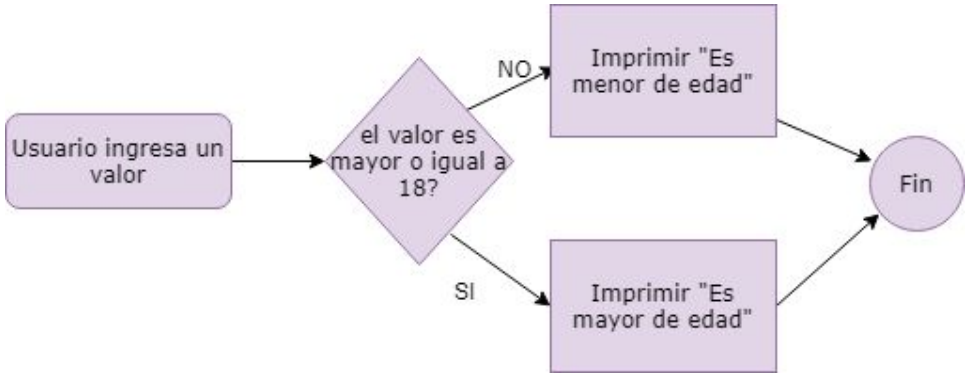


En caso contrario


```
System.out.println("¿Qué edad tienes?");
int edad = sc.nextInt();
if(edad >= 18) {
      System.out.println("Eres mayor de edad");
}
if(edad < 18) {

      System.out.println("Eres menor de edad");
}
```

La instrucción ELSE




```
if(edad >= 18) {
      // código que se ejecutará si se cumple la
condición
      }
else {
      //código que se ejecutará si no se cumple
}
```

Asignación vs comparación

- `=`, es para asignar un valor a una variable
- `==`, es para comparar dos valores o variables

Profundizar en flujo

Objetivo

- Aprender a resolver problemas donde las posibles alternativas a una decisión sean más de dos.
- Analizar situaciones más allá de opción y caso contrario.
- Utilizar la instrucción else if en Java

Planteamiento del problema

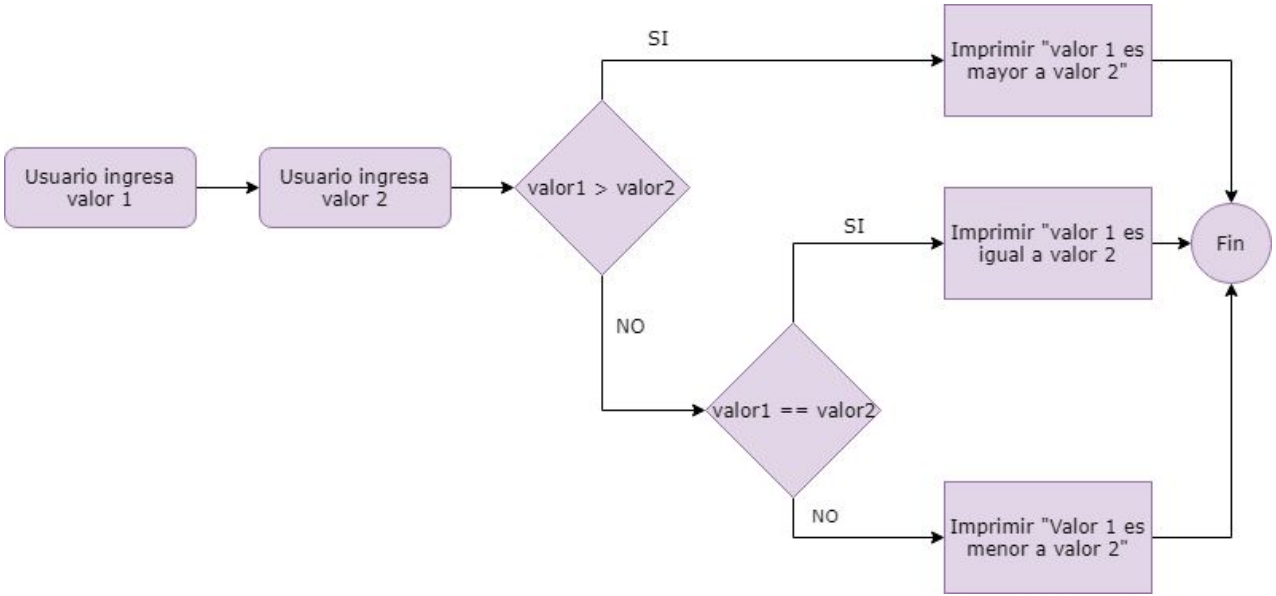
Antes:

- caso 1: A >= B => Decimos que A es mayor
- caso 2: A < B => Decimos que A es menor

Ahora:

- caso 1: A > B => Decimos que A es mayor
- caso 2: A == B => Decimos que A y B son iguales
- caso 3: A < B => Decimos que A es menor que B

Problema el mayor de dos números



Problema el mayor de los números

```
int valor1 = sc.nextInt();
int valor2 = sc.nextInt();
if(valor1 > valor2) {
      System.out.printf("valor1 es mayor a valor2\n");
}
else {
      if(valor1 == valor2) {
            System.out.printf("valor1 es igual a valor2\n");
      }
      else{

            System.out.printf("valor1 es menor a valor2\n");
      }
}
```

Instrucción else if

```
if(valor1>valor2) {
      System.out.printf("valor1 es mayor a valor2\n");
}
else if(valor1 == valor2){
      System.out.printf("valor1 es igual a valor2\n");
}
else{
      System.out.printf("valor1 es menor a valor2\n");
}
```

Clasificación según rangos

Clasificar palabra en corta, mediana y larga:

- 4 letras o menos es corta
- de 5 a 10 letras es mediana
- más de 10 es larga

```
String palabra = sc.nextLine();
int largo = palabra.length();

if(largo <= 4) {
	System.out.printf("Pequeña\n");
}
else if(largo <10){
	System.out.printf("Mediana\n");
}
else{
	System.out.printf("Larga\n");
}
```

Condiciones de borde

Objetivo

- Conocer la importancia de condiciones de borde
- Analizar condiciones de borde
- Evaluar comportamiento de operadores `>`, `>=`, `<` y `<=` en Java

Motivación

- Buscar errores semánticos
- es el tipo de errores más común y el que más tiempo consume
- cuando un problema no funciona como debería, y no hay un error asociado

Analizando una condición de borde

En el ejemplo de mayor de edad.

```
if(edad >= 18) {
      System.out.println("Eres
mayor de edad");
}
else {
      System.out.println("Eres
menor de edad");
}
```

Condición de borde: 18

Analizar para valores 17, 18, 19

Analizando una condición de borde

En el ejemplo de largo de una palabra

```
String palabra = sc.nextLine();
int largo = palabra.length();

if(largo <= 4) {
      System.out.printf("Pequeña\n");
}
else if(largo <10){
      System.out.printf("Mediana\n");
}
else{
      System.out.printf("Larga\n");
}
```

Condiciones de borde: 4 y 10

Analizar para valores 3, 4, 5 y 9, 10, 11

Analizando una condición de borde

En el ejemplo comparando 2 números

```
if(valor1>valor2) {
      System.out.printf("valor1 es mayor a valor2\n");
}
else if(valor1 == valor2){
      System.out.printf("valor1 es igual a valor2\n");
}
else{
      System.out.printf("valor1 es menor a valor2\n");
}
```

Escoger par de números: 2 y 2

Analizar pares 3, 2 y 1, 2

Operadores lógicos

Objetivo

- Hacer uso de los operadores lógicos para evaluar y simplificar expresiones
- Invertir una condición

{desafío} latam_

Motivación

- Ayudan a simplificar flujos

Operadores lógicos

Operador	Nombre	Ejemplo	Resultado
&&	y (and)	false && true	Devuelve true si ambos operadores son true, o ambos false al mismo tiempo. En este ejemplo es false
||	o (or)	false || true	Devuelve true si al menos uno operadores es true, en este ejemplo es true
!	no (not)	!false	Devuelve lo opuesto al resultado de la evaluación, en este caso true

Ejemplo

```
String nombre = "Carlos";
String apellido = "Santana";
boolean a;
a = nombre == "Carlos" && apellido =="Santana";
System.out.println(a);  //true
a = nombre == "Carlos" && apellido =="Vives";
System.out.println(a);  //false
a = nombre == "Carlos" || apellido =="Vives";
System.out.println(a);  //true
```

Identidades

‘Igual” es lo mismo que “no distinto”

```
int numero = 18;
boolean a;
a = numero == 18;
System.out.println(a); //true
a = !(numero != 18);
System.out.println(a); //true
```

Mayor y no menor igual

```
int numero = 18;
boolean a;
a = numero > 18;
System.out.println(a);
a = !(numero <= 18);
System.out.println(a);
```

Simplificando If anidados

Objetivo

- Simplificar problemas con condiciones anidadas en Java
- Entender complicaciones en ifs anidados en Java

Identificando ifs anidados

```
int edad = 30;
boolean zurdo = true;
if(edad >= 18) {
      if(zurdo) {
            System.out.println("Es zurdo y mayor de edad");
      }
}
```

Operadores lógicos

```
if(edad >= 18 && zurdo) {
            System.out.println("Es zurdo y mayor de edad");
}
```

Ejercicio de integración

Crear un programa que solicite 3 números. Determinar el mayor de ellos. (asumir números distintos)

```
System.out.printf("Ingresa el primer numero\n");
int a = sc.nextInt();
System.out.printf("Ingresa el segundo numero\n");
int b = sc.nextInt();
System.out.printf("Ingresa el tercer numero\n");
int c = sc.nextInt();

if(a >= b && a >= c) {
      System.out.printf("a es el mayor");
}
else if(b >= c){
      System.out.printf("b es el mayor");
}
else {
      System.out.printf("c es el mayor");
}
```

Simplificando el flujo

Objetivo

- Simplificar el código de Java utilizando buenas prácticas y la regla de condición en positivo
- Conocer sintaxis de if ternario en Java
- Refactorizar flujo condicional en Java

Variantes del IF: operador terniario

```
si_es_verdadero ? entonces_esto : sino esto;
```

```
System.out.printf("Ingresa el primer numero\n");
int a = sc.nextInt();
System.out.printf("Ingresa el segundo numero\n");
int b = sc.nextInt();
int resultado = (a > b)?a:b;
System.out.printf("%d",resultado);
```

Refactor

```
if(mayorDeEdad == true) {
      if(zurdo == true) {
            System.out.printf("Mayor de edad y zurdo");
      }
      else {
            System.out.printf("Mayor de edad y diestro");
      }
}
else {
      if(zurdo == true) {
            System.out.printf("Menor de edad y zurdo");
      }
      else {
            System.out.printf("Menor de edad y diestro");
      }
}
```

Reemplazando los if anidados por condiciones múltiples

```
boolean mayorDeEdad = true;
boolean zurdo = false;

if(mayorDeEdad == true && zurdo == true) {
            System.out.printf("Mayor de edad y zurdo");}
else if(mayorDeEdad == true && zurdo == false){
      System.out.printf("Mayor de edad y diestro");}
else if(mayorDeEdad == false && zurdo == true){
      System.out.printf("Menor de edad y zurdo");}
else {
      System.out.printf("Menor de edad y diestro");}
```

Análisis Léxico

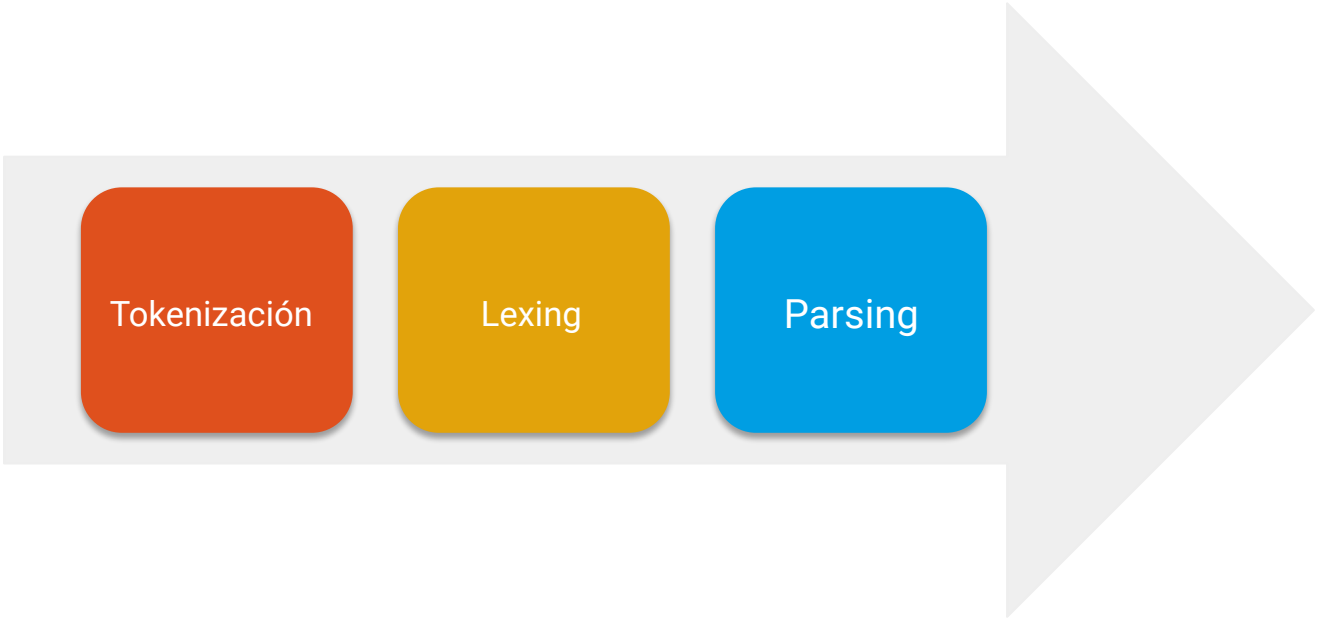
¿Qué sucede cuando ingresamos al terminal y escribimos java miPrograma.java? o ejecutamos desde el IDE

Java lee y procesa el código

No necesitamos saber cómo procesa el código, pero saberlo:

- facilitará nuestra comprensión
- ayudará a entender los mensajes de error

Etapas que realiza Java



Reglas básicas

- Identificadores
- Literales
- Comentarios
- Palabras reservadas

Identificadores

- nombres de variables
- nombres de clases
- nombres de métodos

- comenzar con letra de a - z
- comenzar con _ o $
- puede tener números
- no puede empezar con números

Por ejemplo, si colocamos aparecerá el siguiente error

```
int 1a = 3;
```

```
Exception in thread "main" java.lang.Error:
Unresolved compilation problem:
      Syntax error on token "1", delete this token
```

Literales

```
String nombre = "Chuck Berry";
```

valores que asignamos a las variables o simplemente utilizamos para operar

Comentarios

```
// Esta línea es un comentario
// Los comentarios son ignorados
a = 3 * 5; // también pueden existir contiguos a
líneas de código válido
```

Palabras reservadas

abstract	do	implements	protected	throw
boolean	double	import	public	throws
break	else	instanceof	rest	transient
byte	extends	int	return	true
case	false	interface	short	try
catch	final	long	static	void
char	finally	native	strictfp	volatile
class	float	new	super	while
const*	for	null	switch	
continue	goto*	package	synchronized	
default	if	private	this	

Quizz

{desafío}
latam_

Cierre

{desafío}
latam_

Reflexionemos

¿Hay algún contenido que aún no tengas la seguridad de haberlo aprendido totalmente?

{desafío} latam_

{desafío} latam_

Academia de talentos digitales