



# Fundamentos de Java



Módulo 1 - Clase #02  
Java Programmer SE 8



# Manipulación de Cadenas: Clase String

---

- Se almacena con una instancia.
- Son inmutables.
- Métodos principales:
  - `length`: devuelve la cantidad de caracteres de la cadena.
  - `toUpperCase`: devuelve la cadena convertida a mayúsculas.
  - `toLowerCase`: devuelve la cadena convertida a minúsculas.
  - `equals`: compara dos cadenas y devuelve true si son iguales.

## **Instancia:**

```
String cadena = "Hola mundo";  
String blog = "Picando";  
blog += " ";  
blog += "Código";  
System.out.println(blog);
```

## **Método:**

```
public String toString() {  
    int longitud = cadena.length;
```

# Construcción de Cadenas: String vs StringBuffer vs StringBuilder

---

- Se debe entender que:
  - Son inmutables.
  - Las cadenas de caracteres son arreglos de caracteres.
- Diferencias entre clases:
  - String no permite cambiar el valor de la cadena de caracteres, es síncrona.
  - StringBuilder permite cambiar la cadena de caracteres y es síncrona.
  - StringBuffer permite cambiar la cadena de caracteres, es síncrona y además es multihilo.

# Ejemplo



# Leer datos por teclado: Scanner

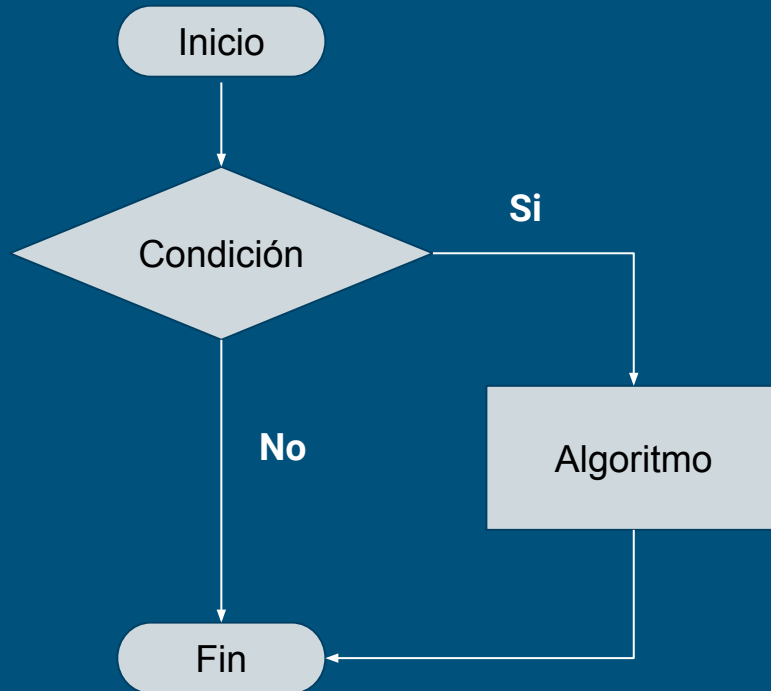
---

- Aclaración:
  - `System.out`
  - `System.in`
- Clase:
  - `java.util.Scanner`
- Es un tipo de dato object.
- Está diseñada para leer los bytes y convertirlo en valores primitivos (int, double, bool, etc) o en valores String
- Entre sus métodos principales:
  - `nextByte()`
  - `nextDouble()`
  - `nextFloat()`
  - `nextInt()`
  - `next()`
  - `nextLine()`
  - `nextLong()`

# Ejemplo



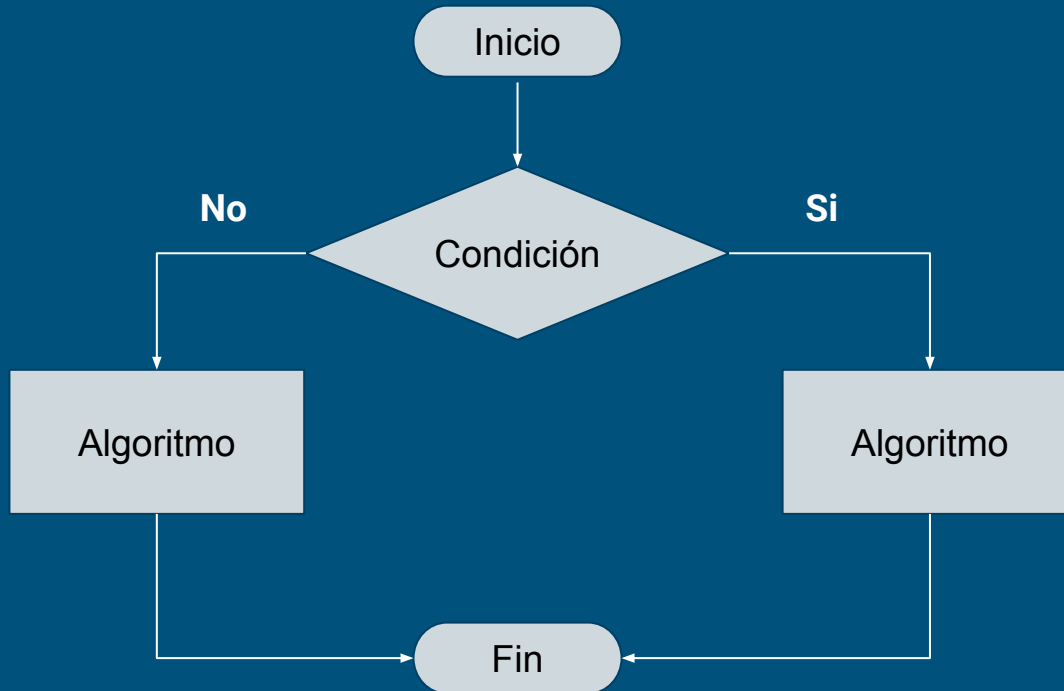
# Estructura de Control: Condicionales



## Simple:

- Evalúa la condición.
- Ejecuta una serie de instrucciones siempre y cuando la condición se cumpla.
- Si no se cumple, continúa con el resto del algoritmo.

# Estructura de Control: Condicionales

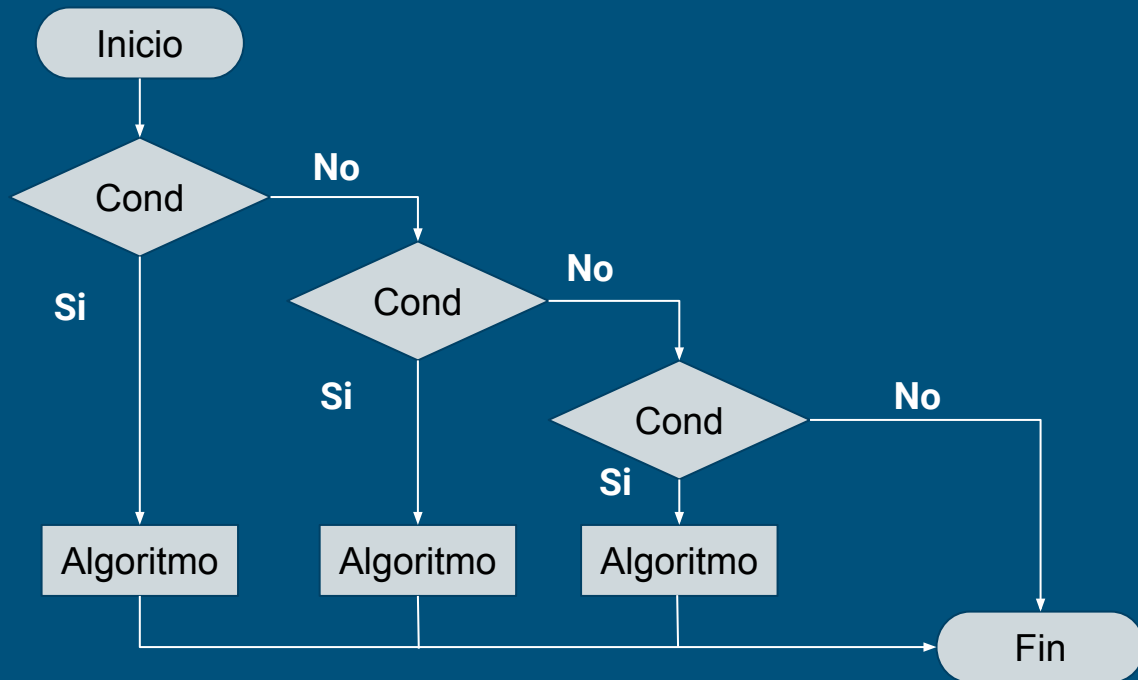


## Doble:

- Evalúa la condición.
- Ejecuta una serie de instrucciones siempre y cuando la condición se cumpla.
- Si no se cumple, ejecuta otra serie de instrucciones.



# Estructura de Control: Condicionales



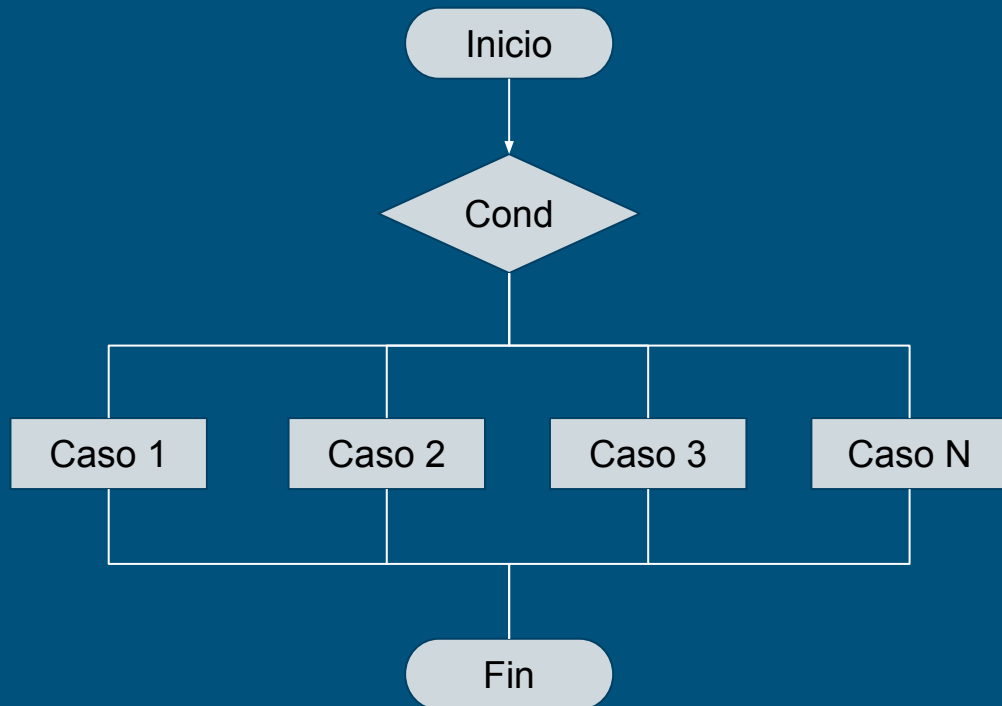
## Compuesto:

- Evalúa la condición.
- Ejecuta una serie de instrucciones siempre y cuando la condición se cumpla.
- Si no se cumple, evalúa otra condición y ejecuta otra serie de instrucciones en caso de cumplirse.
- Se puede repetir N cantidad de veces.

# Ejemplo



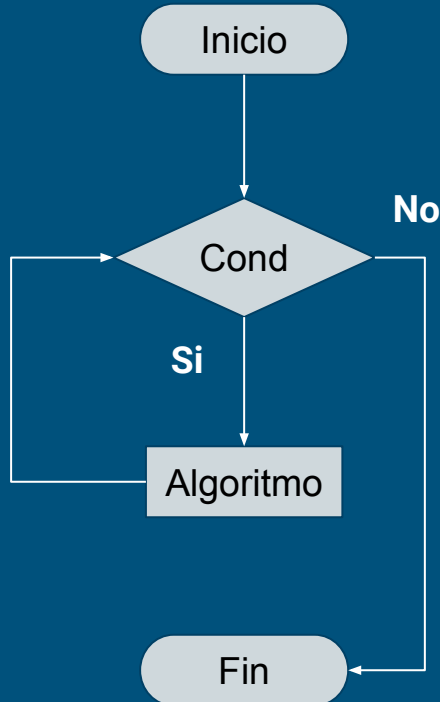
# Estructura de Control: Switch



## Funcionamiento:

- Evalúa la condición.
- Posee casos que la condición debe cumplir.
- Si la condición no cumple con alguno de los casos ejecuta uno por defecto.

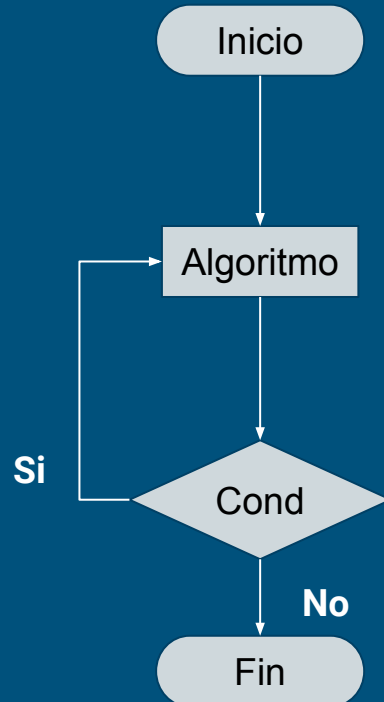
# Estructuras Iterativas: While



## Funcionamiento:

- Evalúa la condición.
- Si se cumple ejecuta el algoritmo y vuelve a evaluar la condición. Este paso se repite N cantidad de veces
- Si la condición no se cumple sale del ciclo.

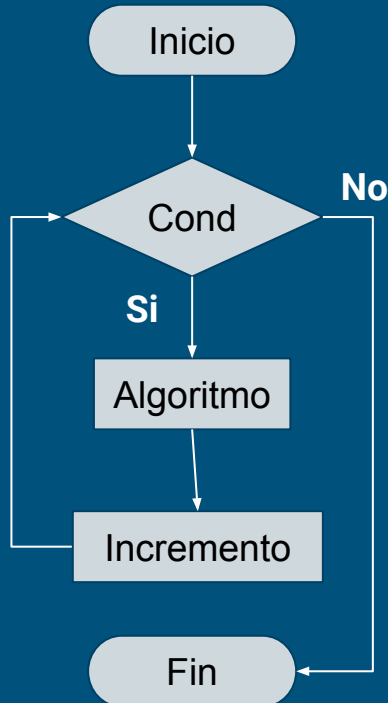
# Estructuras Iterativas: Do-While



## Funcionamiento:

- Ejecuta el algoritmo al menos una vez.
- Evalúa la condición.
- Si se cumple vuelve a ejecutar el algoritmo y luego evalúa la condición de nuevo. Este paso se repite N cantidad de veces
- Si la condición no se cumple sale del ciclo.

# Estructuras Iterativas: For



## Funcionamiento:

- Establece los parámetros de inicio y fin del ciclo.
- Evalúa la condición.
- Ejecuta el algoritmo.
- Incrementa/decrementa el indicador de iteración del ciclo.
- Si se cumple vuelve a ejecutar el algoritmo. Se repiten los pasos hasta que la condición no se cumpla.

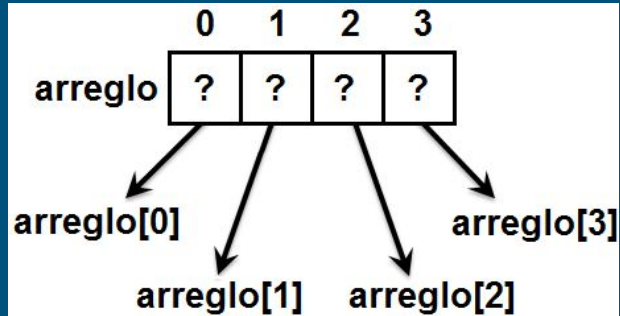
# Ejemplo



# Arreglos

Un arreglo puede definirse como un grupo o una colección finita, homogénea y ordenada de elementos. Los arreglos pueden ser:

Unidimensionales (Vectores).



Bidimensionales (Matrices).

A diagram illustrating a 2D array (matrix). It shows a 3x3 grid of cells. Above the grid is a red bracket spanning all three columns, with the label 'columnas' centered above it. To the left of the grid is a red bracket spanning all three rows, with the label 'filas' centered to its left. The values in the cells are as follows:

10	-3	4
6	7	-2
14	48	-33



# ArrayList

Los ArrayList forman parte de la API Collection de Java (una serie de clases para manejo de datos). Nos permiten procesar con información de una manera parecida a los arreglos, la diferencia es que podemos agregar elementos dinámicamente sin necesidad de definir un tamaño.

Además, la forma de agregar elementos es diferente a los arreglos ya que utiliza métodos para esto. Además se pueden buscar elementos de manera más rápida y podemos manipular sus elementos de mejor forma que con los arreglos.

Elemento 0

Elemento 1

Elemento 2

Elemento 3

Elemento 4

Elemento 5



# Métodos para ArrayList

---

```
ArrayList<String> al = new ArrayList<String>();
```

```
// Añade el elemento al ArrayList
```

```
al.add("Elemento");
```

```
// Devuelve el número de elementos actuales
```

```
al.size();
```

```
// Devuelve el elemento en la posición '2'
```

```
al.get(2);
```

```
// Buscar
```

```
al.contains("Elemento");
```

```
// Borrar
```

```
al.remove(5);
```

```
// Borrar todos los elementos
```

```
al.clear();
```

```
// Conocer si está vacío o no
```

```
al.isEmpty();
```

# Ejemplo

