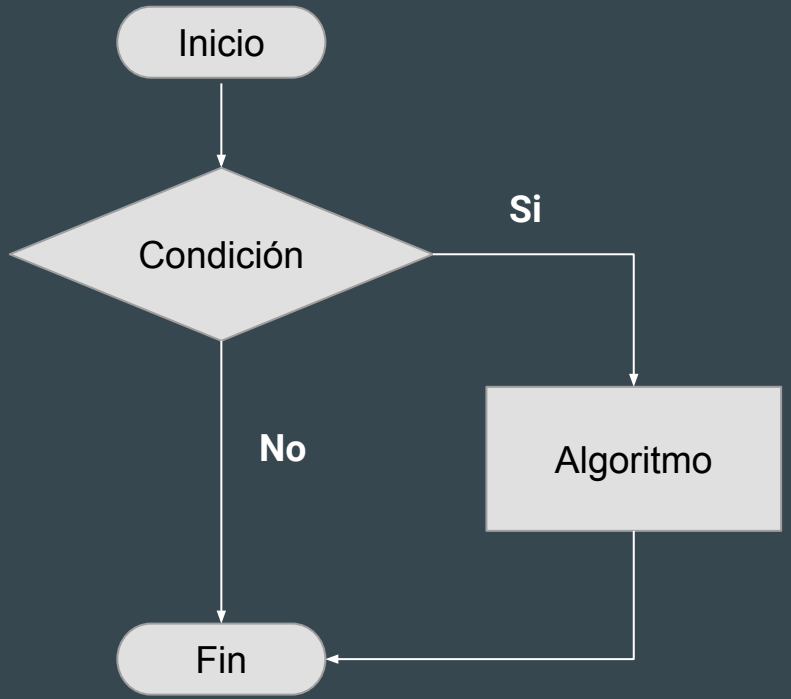
Fundamentos de Java

Estructuras de Control, Iterativas y Datos

Contenido

- Estructuras de Control:
  - Condicionales:
    - if - else - elseif
  - Casos:
    - switch
- Estructuras Iterativas (Ciclos):
  - For
  - ForEach
  - While
  - Do-While
- Arreglos:
  - Unidimensionales (vectores)
  - Bidimensionales (matrices)
- Listas de Arreglos:
  - ArrayList

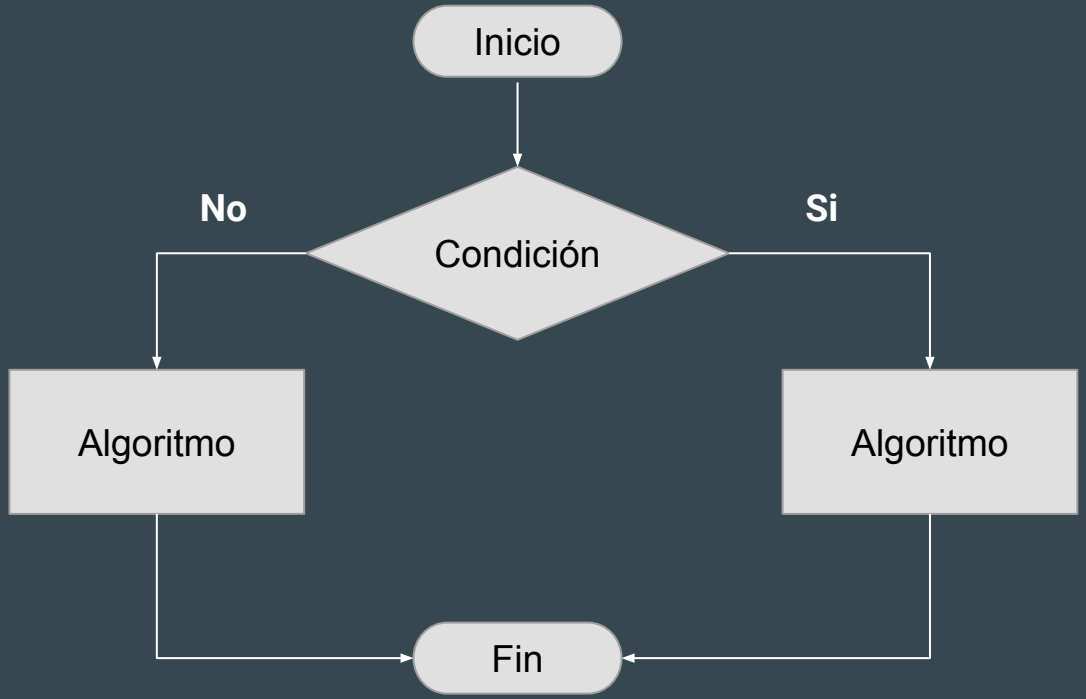
Estructura de Control: Condicionales



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



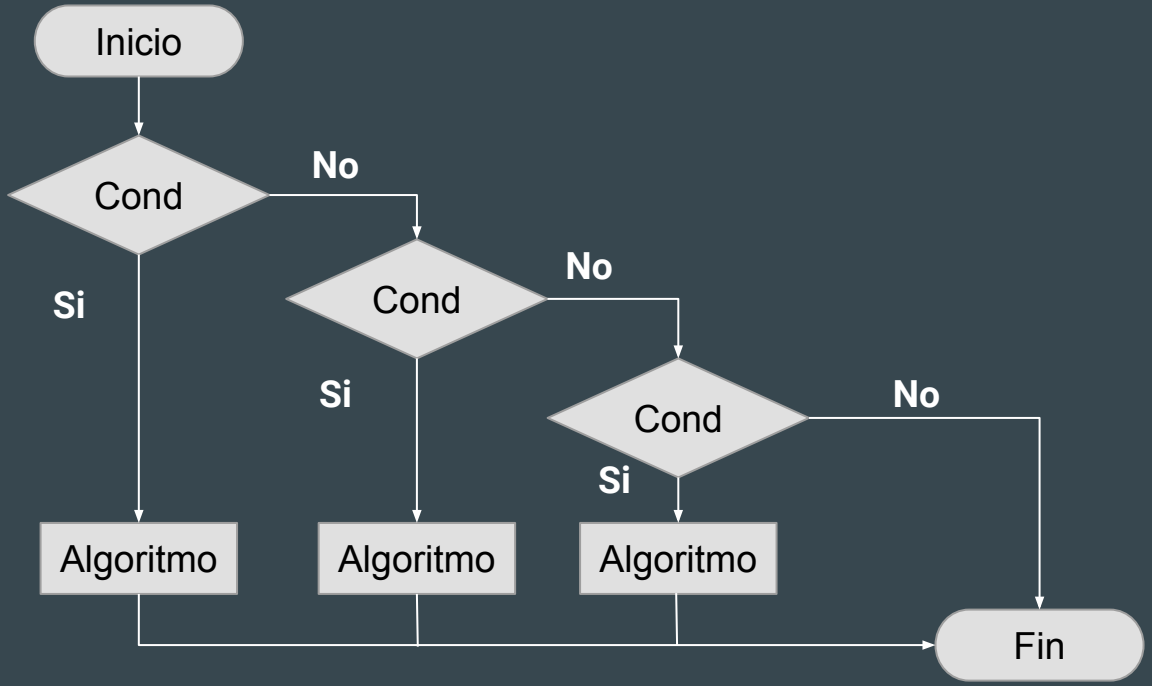
Doble:

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

Estructuras de Control: Condicionales

```
public class Example {

    public static void main(String[] args) {
        // Condicional simple
        if((3*3) == 9) { // 9 es igual 9
            System.out.println("SÍ Se cumplió la condición.");
        }
        // Condicional doble
        if((3*3) == 5) { // 9 no es igal a 5
            System.out.println("SÍ Se cumplió la condición.");
        } else {
            System.out.println("NO se cumplió la condición.");
        }
    }

}
```

Estructura de Control: Condicionales



Compuesto:

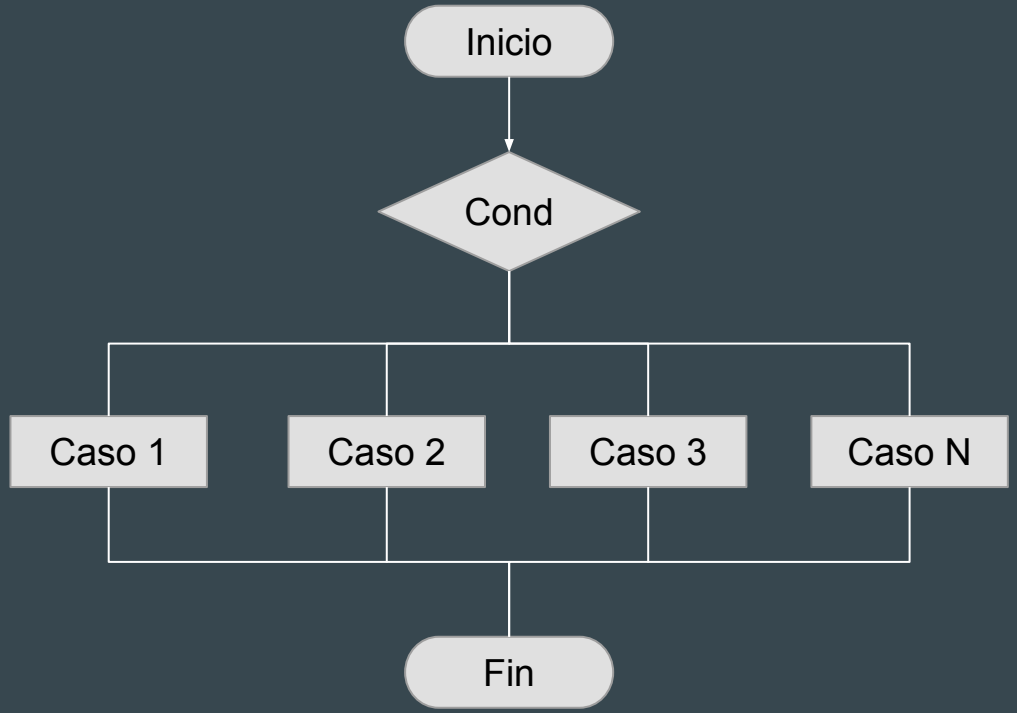
- Evalúa la condición.
- Ejecuta una serie de instrucciones siempre y cuando la condición se cumpla.
- Si no se cumple, evalúa otra condición y ejecuta otra serie de instrucciones en caso de cumplirse.
- Se puede repetir N cantidad de veces.

Estructuras de Control: Condicionales

```java
public class Example {

    public static void main(String[] args) {
        int option = 3;
        // Condicional multiple
        if(option == 1) {
            System.out.println("Ha escogido la opción #1.");
        } else if(option == 2) {
            System.out.println("Ha escogido la opción #2.");
        } else if(option == 3) {
            System.out.println("Ha escogido la opción #3.");
        } else {
            System.out.println("Ninguna de las anteriores.");
        }
    }
}
```

Ejemplo

Estructura de Control: Switch



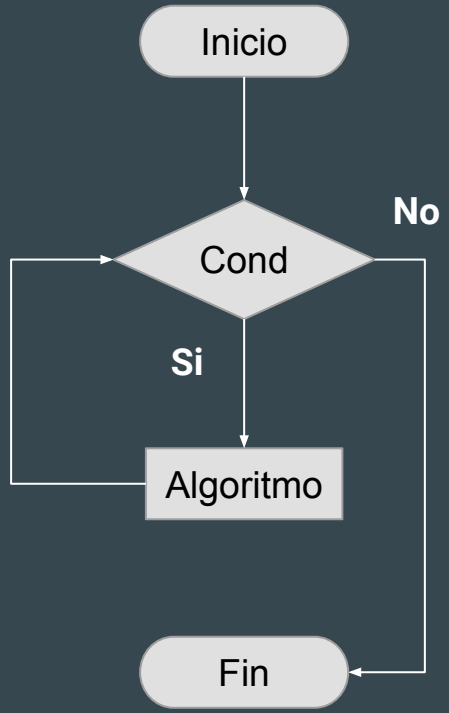
Funcionamiento:

- 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 de Control: Switch

```java
public class Example {

    public static void main(String[] args) {
        int option = 3;
        // Switch case
        switch (option) {
            case 1:
                System.out.println("Ha escogido la opción #1.");
                break;
            case 2:
                System.out.println("Ha escogido la opción #2.");
                break;
            case 3:
                System.out.println("Ha escogido la opción #3.");
                break;
            default:
                System.out.println("Ninguna de las anteriores.");
                break;
        }
    }
}
```

Estructuras Iterativas: While



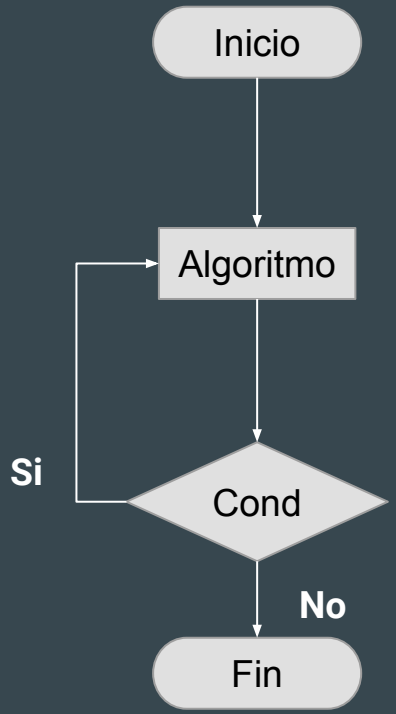
Funcionamiento:

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

Estructuras Iterativas: While

```java
public class Example {

    public static void main(String[] args) {
        boolean verify = true;
        int i = 0;

        // Ciclo While
        while(verify) {
            i++;
            System.out.println("2 * " + i + " = " + (2*i));
            if(i == 10) {
                verify = false;
            }
        }
    }
}
```

Estructuras Iterativas: Do-While



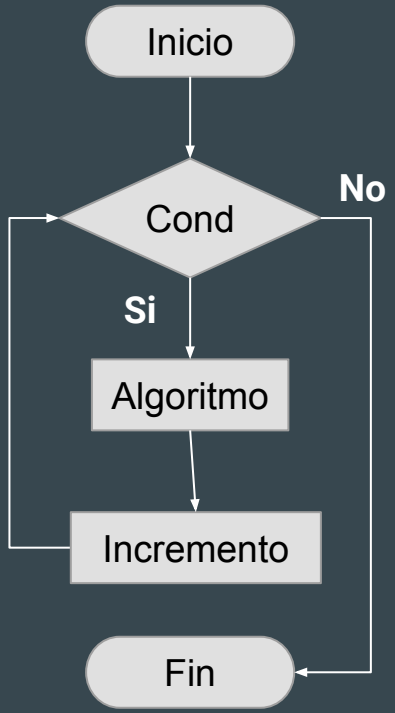
Funcionamiento:

- 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: Do-While

```java
public class Example {

    public static void main(String[] args) {
        boolean verify = true;
        int i = 0;

        // Ciclo DoWhile
        do {
            i++;
            System.out.println("2 * " + i + " = " + (2*i));
            if(i == 10) {
                verify = false;
            }
        } while(verify);
    }
}
```

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.

Estructuras Iterativas: For

```java
public class Example {

    public static void main(String[] args) {
        // Ciclo For
        for(int i = 1; i <= 10; i++) {
            System.out.println("2 * " + i + " = " + (2*i));
        }
    }
}
```

Estructuras Iterativas: For

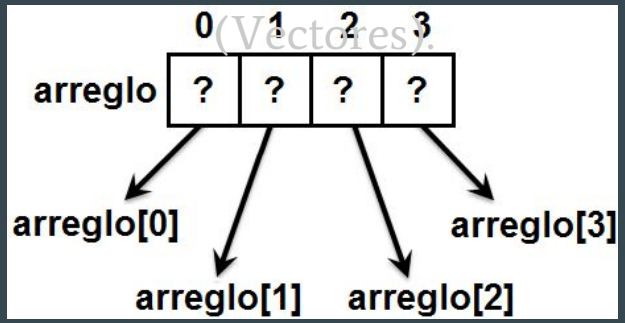
```java
public class Example {

    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
        // Ciclo Foreach
        for(int n : numbers) {
            System.out.println("2 * " + n + " = " + (2*n));
        }
    }
}
```

Ejemplo

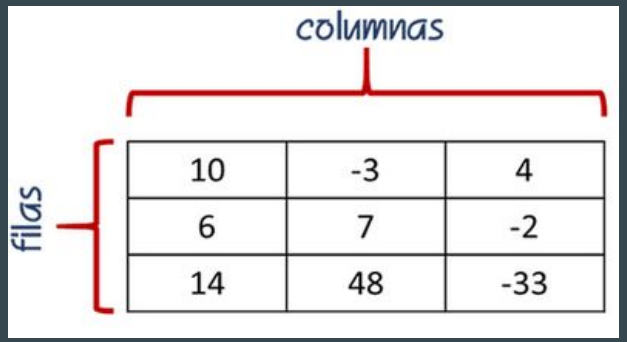
Arreglos

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

Unidimensionales (Vectores):



Bidimensionales (Matrices)



Arreglos

```java
public class Example {

    public static void main(String[] args) {
        // Arreglos unidimensionales (Vectores)
        // Declaración
        String[] students;
        // inicialización
        students = new String[5];
        // Declaración e inicialización
        int[] numbers = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; // Asignación manual
        // Asignación por manual indice de posición
        students[0] = "Luis Ochoa";
        students[1] = "Sofia Vargas";
        students[2] = "Carlos Pérez";
        students[3] = "Celeste Rodriguez";
        students[4] = "Erika López";
        // Muestra
        System.out.println(students[3]);// Celeste Rodríguez
        // Para llenar con ciclos
        System.out.println(numbers.length);//10 - cantidad de posiciones.
    }
}
```

Arreglos

```java
public class Example {
    public static void main(String[] args) {
        // Arreglos bidimensionales (Matrices)
        // Declaración
        String[][] codes;
        // inicialización
        codes = new String[2][2];
        // Declaración e inicialización
        int[][] tables = {
       /*0*/{1, 2, 3, 4, 5},// 00, 01, 02, 03, 04
       /*1*/{2, 4, 6, 8, 10},// 10, 11, 12, 13, 14
       /*2*/{3, 6, 9, 12, 15}// 20, 21, 22, 23, 24
        }; // Asignación manual
        // Asignación por manual índice de posición
        codes[0][0] = "JUY";
        codes[0][1] = "ADW";
        codes[1][0]= "XVN";
        codes[1][1] = "PLH";
        // Muestra
        System.out.println(codes[0][1]);// ADW
        // Para llenar con ciclos
        System.out.println(codes[0].length);// 2 - cantidad de posiciones.
    }
}
```

ArrayList

- Forman parte de la colección de clases (API) de Java
- Nos permite procesar la información de una manera parecida a los arreglos
- Podemos agregar elementos dinámicamente sin necesidad de definir un tamaño
- Se utilizan métodos para agregar elementos
- Se pueden buscar elementos más rápido y manipular elementos de mejor forma que con los arreglos

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