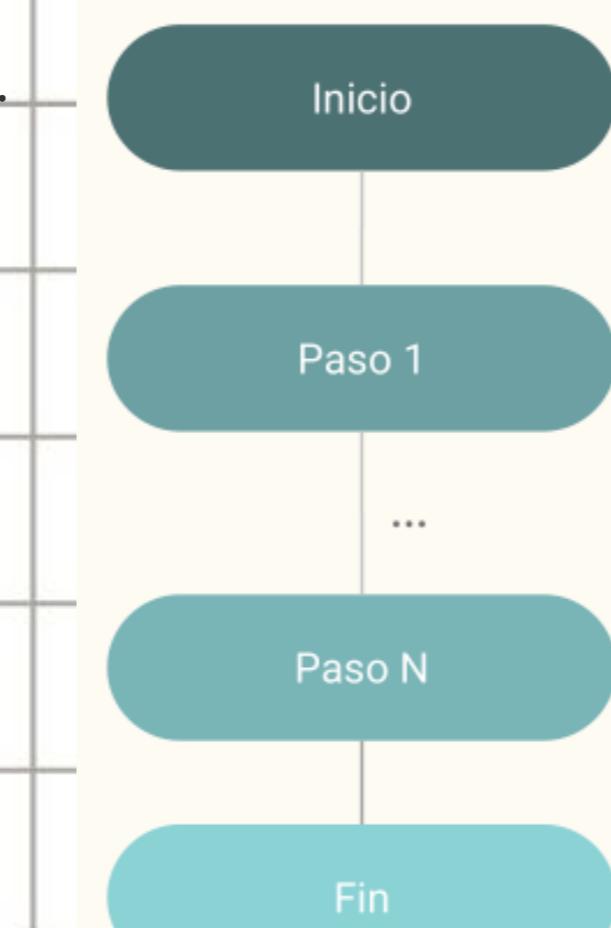
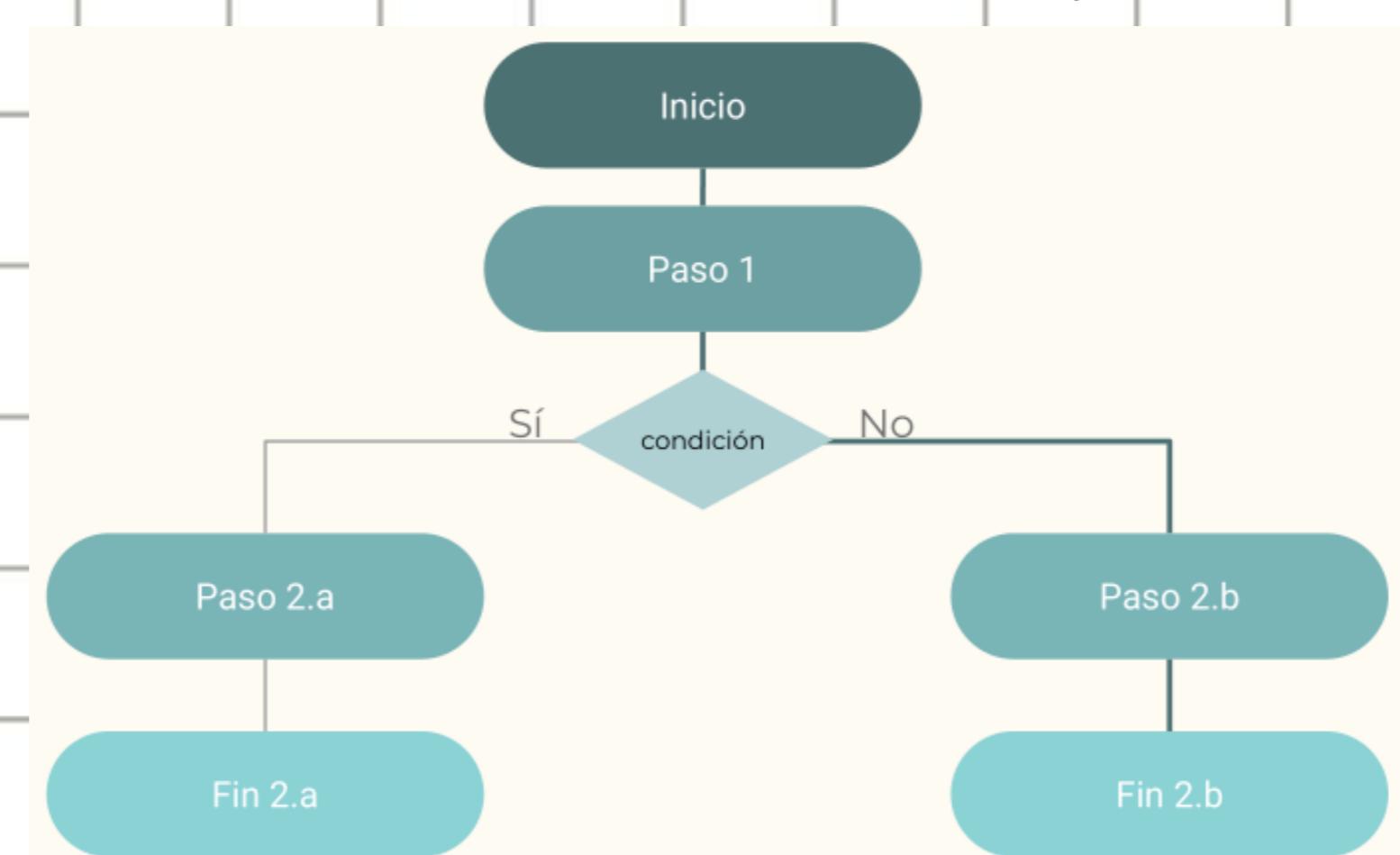


Estructuras - Entrada / salida - archivosEstructuras algorítmicasSecuenciales - condicionales - repetitivas.Estructuras algorítmicas secuenciales

- Son estructuras que desde su inicio hasta su fin siguen un orden fijo, tal cual está escrito.

Estructuras algorítmicas condicionales

- Son estructuras que pueden variar el flujo que siguen según si se cumple o no la condición. Sigue solo uno de los caminos.



IF Permite ejecutar un bloque de código si se cumple una condición.

```

if(edad >= MAYORIA_EDAD){
    System.out.println("Es mayor de edad");
}
  
```

If - else if - else

```
if(edad >= MAYORIA_EDAD){  
    System.out.println("Es mayor de edad");  
}else if (edad > INICIO_ADOLESCENCIA){  
    System.out.println("Es adolescente");  
}else if (edad > MINIMA_EDAD){  
    System.out.println("Es un infante");  
}else{  
    System.out.println("No es valida la edad: " + edad);  
}
```

Operador ternario

No es un operador tan usado ya que dificulta la comprensión del código.

Se usa para casos condicionales simples del estilo si - no.

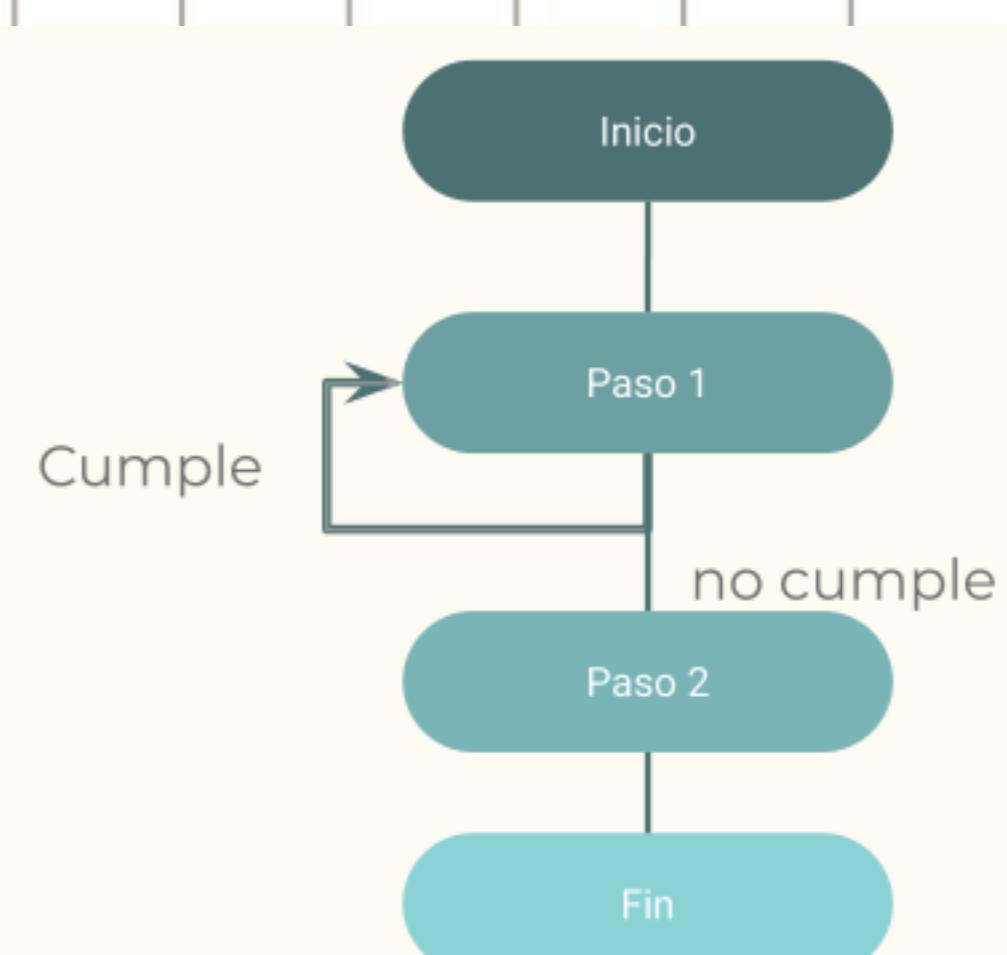
```
String mensaje = edad >= MAYORIA_EDAD ? "Es mayor de edad" : "No es mayor de edad";
```

Switch

```
switch (opcionElegida) {  
    case AGREGAR:  
        agregarCalificacion();  
        break;  
    case BUSCAR:  
        buscarCalificacion();  
        break;  
    case ELIMINAR:  
        eliminarCalificacion();  
        break;  
    case SALIR:  
        System.out.println("¡Hasta luego!");  
        break;  
    default:  
        System.out.println("Opción no válida. Intenta nuevamente.");  
}
```

Estructuras algorítmicas repetitivas

- Son estructuras en las que se repite un paso hasta que no se cumpla la condición. Eventualmente se tiene que dejar de cumplir sino me encuentro en un loop infinito.



for Se usa cuando se la CANTIDAD EXACTA de iteraciones a realizar. No debo cortar el flujo antes. Es el único que se permite declarar variables con caracteres "i", "j", "k", etc.

```
for (int i = 1; i <= 5; i++) {
    System.out.println("Número: " + i);
}
```

for each Se usa cuando quiero recorrer todos los elementos de una estructura. No debo cortar el flujo antes.

```
for (int numero : numeros) {
    System.out.println("Número: " + numero);
}
```

while Se usa cuando no se la cantidad exacta sino que quiero recorrer hasta que se cumpla x condición.
No debo cortar el flujo con algo que no sea la condición de corte. Puede no ejecutarse ninguna vez.

```
while (!encontrado && contador < numeros.length) {
    if (numeros[contador] == numeroBuscado) {
        encontrado = true;
    }
    contador++;
}
```

do-while Se usa cuando se que quiero ejecutar el código mínimo una vez y hasta que se cumpla la condición de corte.

```
do {
    var contrasenia = pedirContrasenia();
    esValida = esContraseniaValida(contrasenia);
    if (!esValida) {
        System.out.println("Contraseña incorrecta. Intenta nuevamente.");
    }
} while (!esValida);
```

Entrada / Salida de datos

Entrada Para pedir datos al usuario vamos a usar Scanner una clase de la biblioteca `java.util`. El mismo se instancia como:

```
Scanner scanner = new Scanner(System.in);
```

y tiene métodos que nos permiten leer líneas de texto enteras como:

```
scanner.nextLine()
```

O pedir un número:

```
scanner.nextInt() / scanner.nextDouble()
```

No olvidarse de cerrar el scanner al terminar de utilizarlo.

Salida Para mostrarle datos al usuario vamos a usar System.out el cual se usa como:

```
System.out.println("La opción seleccionada no se encuentra entre las posibles");
```

También podemos agregar variables al `println`:

```
System.out.printf("El resultado de la suma es: %s%n", resultado);
```

```
System.out.printf("El resultado de la suma es: " + resultado);
```

Ejercicio Armar un menú que se tiene que mostrar por lo menos una vez para una calculadora. Debe tener las opciones

- Sumar → pedir los 2 números al usuario

- Restar → "

- Salir → decir adios.

```
do {  
    menu.mostrarMenu();  
    opcion = menu.pedirOpcionValida();  
    menu.procesarOpcion(opcion, calculadora);  
} while (!menu.esOpcionSalir());
```

```
/**  
 * Muestra las opciones disponibles en el menú  
 */  
public void mostrarMenu() {  
    System.out.println("Menú de Calculadora:");  
    System.out.println("1. Sumar");  
    System.out.println("2. Restar");  
    System.out.println("3. Mostrar historial");  
    System.out.println("4. Salir");  
}
```

ArchivosTipos de archivos

.TXT

.CSV

.BIN

Texto

Separado
por coma

Binario

hola como estas

hola,como,estas

01101000 01101111
01101100 01100001....

Apertura Un archivo de texto se puede abrir para lectura o para escritura y en este último caso se puede iniciar escribiendo desde 0 o agregando (append) al final.

```

try {
    FileReader archivo = new FileReader(ruta);
    //Codigo
} catch (IOException e) {
    throw new RuntimeException(e);
}

try {
    FileWriter fileWriter = new FileWriter(fileName: "test.txt");
} catch (IOException e) {
    throw new RuntimeException(e);
}

try (FileWriter archivo = new FileWriter(ruta, append: true)) {

```

Para archivos binarios se debe utilizar FileInputStream y

FileOutputStream en lugar de FileReader y FileWriter.

Chequeo de apertura Si no se puede abrir se va a lanzar una excepción la misma se debe manejar como se crea conveniente, si no se puede seguir con la ejecución debe cerrarse el programa.

Lectura Un archivo de texto se puede leer con cualquiera de los siguientes métodos :

<code>read()</code>	Reads a single character.
<code>read(char[] cbuf)</code>	Reads characters into an array.
<code>read(char[] cbuf, int off, int len)</code>	Reads characters into a portion of an array.

También se Puede aprovechar lo visto con Scanner Para facilitar la lectura .

```
Scanner scanner = new Scanner(archivo);
while (scanner.hasNextLine()) {
    String linea = scanner.nextLine().trim();
    //hago cosas
}
```

Escritura Un archivo de texto se puede escribir utilizando cualquiera de los siguientes métodos según lo necesario :

<code>write(char[] cbuf)</code>	Writes an array of characters.
<code>write(char[] cbuf, int off, int len)</code>	Writes a portion of an array of characters.
<code>write(int c)</code>	Writes a single character.
<code>write(String str)</code>	Writes a string.
<code>write(String str, int off, int len)</code>	Writes a portion of a string.

Cierre Siempre que terminamos de usar un archivo debemos recordar cerrarlo con : `file.close();`

El archivo debe cerrarse en el mismo scope que se abrió .

Ejercicio Al salir del Programa de la calculadora agregue al historial.txt los resultados de las cuentas realizadas.

historial.txt tiene un formato: número

número

número

No se debe perder el historial al iniciar otra vez la calculadora.

ALGORITMOS Y ESTRUCTURAS DE DATOS

```
public class Main {  
    public static void main(String[] args) {  
        Calculadora calculadora = new Calculadora();  
        int opcion;  
        Historial gestorHistorial = new Historial();  
        Menu menu = new Menu(gestorHistorial);  
  
        do {  
            menu.mostrarMenu();  
            opcion = menu.pedirOpcionValida();  
            menu.procesarOpcion(opcion, calculadora);  
        } while (!menu.esOpcionSalir());  
  
        gestorHistorial.guardarHistorial();  
    }  
}
```

```
/**  
 * Se encarga del flujo de sumar  
 */  
private void sumar(Calculadora calculadora) { 1 usage  
    var primerNumero = pedirNumero( mensaje: "Ingresa el primer número: ");  
    var segundoNumero = pedirNumero( mensaje: "Ingresa el segundo número: ");  
    var resultado = calculadora.sumar(new int[]{primerNumero, segundoNumero});  
    historial.guardarResultado(resultado);  
    System.out.printf("El resultado de la suma es: %s%n", resultado);  
}
```

```
/**  
 * Guarda el resultado en el historial  
 */  
public void guardarResultado(int resultado){ 2 usages  
    historialEjecucion.add(resultado);  
}
```

```
/**  
 * Se encarga de agregar los resultados al archivo del historial  
 * @param resultados resultados a guardar, es importante no reenviar los ya guardados  
 */  
public void guardarHistorial(ArrayList<Integer> resultados) { 1 usage  
    System.out.println("Escribiendo historial en: " + new java.io.File(ruta).getAbsolutePath());  
    if(!resultados.isEmpty()){  
        try (FileWriter archivo = new FileWriter(ruta, append: true)) {  
            for (int resultado : resultados) {  
                archivo.write(str: "\n" + resultado);  
            }  
        } catch (IOException e) {  
            System.out.println("Error al escribir un número en el archivo de historial ");  
        }  
    }  
}
```