Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

| Nombre Apellido: | | NOTA | |
|---|---|---|---|
| Módulo: | Entornos de desarrollo | Fecha: | 11/05/2025 |
| C.F: | DAM_SM | Curso: | 1º |
| Profesorado: | Joan Agustí Suàrez | | |
| UD3 y UD4: | Pruebas, refactorización y JavaDoc | | 11:15-13:10 |

**NOTA: En el bloque examen tercera evaluación (esta vez no es terrible) tenéis disponible el fichero .java del que se habla a continuación, además la tarea donde entregar los documentos que generéis.**

```java
public class Calculadora {

    public int operar(String operador, int a, int b) {
        if (operador.equals("sumar")) {
            System.out.println("🔹 Iniciando operación: SUMAR");
            int resultado = a + b;
            System.out.println("Resultado: " + resultado);
            return resultado;
        } else if (operador.equals("restar")) {
            System.out.println("🔹 Iniciando operación: RESTAR");
            int resultado = a - b;
            System.out.println("Resultado: " + resultado);
            return resultado;
        } else if (operador.equals("multiplicar")) {
            System.out.println("🔹 Iniciando operación: MULTIPLICAR");
            int resultado = a * b;
            System.out.println("Resultado: " + resultado);
            return resultado;
        } else if (operador.equals("dividir")) {
            System.out.println("🔹 Iniciando operación: DIVIDIR");
            if (b == 0) {
                System.out.println("⚠ Error: División por cero");
                throw new ArithmeticException("División por cero");
            }
            int resultado = a / b;
            System.out.println("Resultado: " + resultado);
            return resultado;
        } else {
            System.out.println("⚠ Error: Operación no válida");
            return 0;
        }
    }
}
```

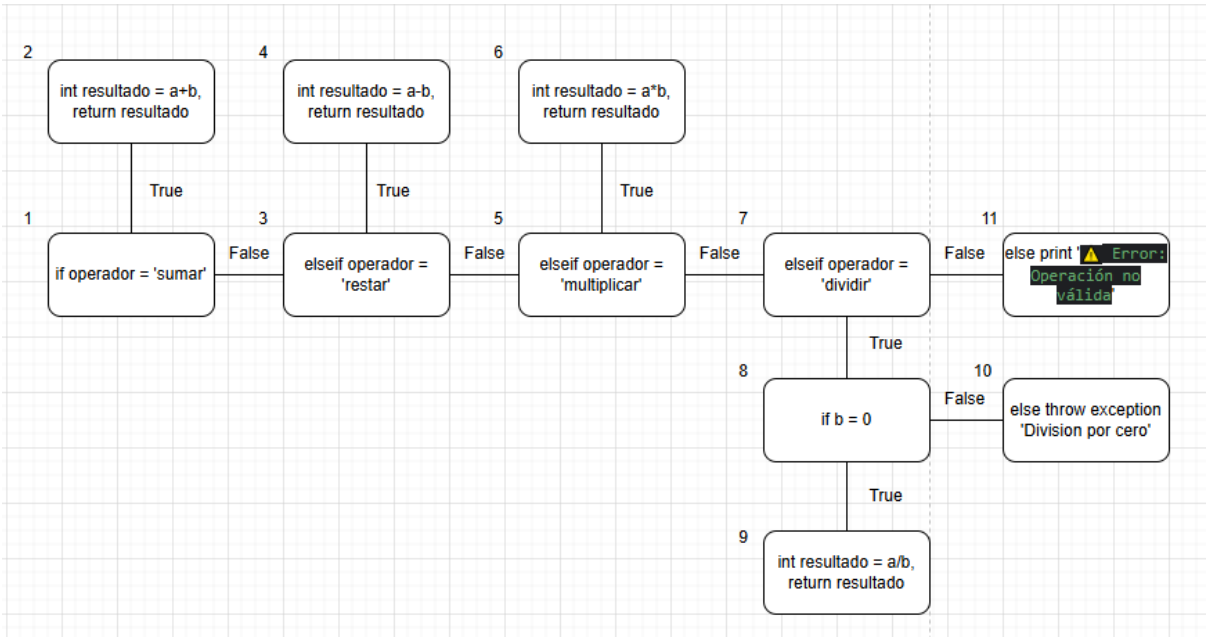
Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

**Actividad 1. Caja negra.** (La cantidad de filas es orientativa, no quiere decir que tengáis que hacer 7)

| Prueba | Resultado esperado | Resultado obtenido | Conclusión |
|---|---|---|---|
| Sumar (1, 1) | 2 | 2 | correcto |
| Sumar (-3, -2) | -5 | -5 | Correcto |
| Restar (4, -3) | 7 | 7 | correcto |
| Multiplicar (123, 0) | 0 | 0 | correcto |
| Dividir (213, 0) | 'Error: division por cero' | 'Error: division por cero' | correcto |
| | | | |
| | | | |

**Actividad 2. Caja Blanca.** Dibuja el grafo (diagrama de nodos como algunos le llamáis) marcando claramente los nodos predicado y calcula la complejidad ciclomática.

Rellena una tabla con los diferentes caminos que pueden existir.



| | | | |
|---|---|---|---|
| Camino 1, 2 | int resultado = a+b, return resultado | | |
| Camino 1, 3, 4 | int resultado = a-b, return resultado | | |
| Camino 1, 3, 5, 6 | int resultado = a*b, return resultado | | |
| Camino 1, 3, 5, 7, 8, 10 | throw exception 'Division por cero' | | |
| Camino 1, 3, 5, 7, 8, 9 | int resultado = a/b, return resultado | | |
| Camino 1, 3, 5, 7, 8, 11 | Print '⚠ Error: Operación no válida' | | |

**Actividad 3. JUnit.** Realiza las pruebas unitarias necesarias para cubrir el 100% de este código.

```
/*@Test
void null
Calculadora = new Calculadora;
int resultado = null;
int a = null;
int b = null;
*/
```

**Actividad 4.** Aplica todos los patrones de **refactorización** que detectes. Enumera los patrones que has utilizado y el porqué.

```
public int operarRefactorizado(String operador, int a, int b) {
    System.out.println(" Iniciando operación: " + operador.toUpperCase());
    switch (operador){
        case "sumar":
            return a + b;
        case "restar":
            return a - b;
        case "multiplicar":
            return a * b;
        case "dividir":
            if (b == 0) {
                throw new ArithmeticException("División por cero");
            }
            return a / b;
        case null, default:
            System.out.println("⚠ Error: Operación no válida");
            return 0;
    }
}
```

eliminado prints(“Iniciando operación:  (operación en cuestion)”) → reemplazado por print(“Iniciando operación:  ” + operador) al principio del método ya que de esta forma ocupa 1 linea de código y no 4

eliminado ‘resultado’ → parametro rebundante y ocupa lineas, reemplazado por la operación en sí en el return

eliminado ‘print(‘resultado: ’ + resultado)’ → print rebundante que se repite 4 veces y se puede eliminar porque ya esta en el main

eliminado ‘print(" Error: División por cero")’ → rebundante ya que la excepcion de la siguiente linea hace lo mismo
switch en vez de ifs y elseifs → hacer tantos ifs en un código es menos refinado y ocupa más código que un switch

**Actividad 5. Documenta** el código que has obtenido lo máximo que puedas e indica a continuación los pasos a seguir si queremos generar el JavaDoc.

```
/* Este metodo realiza una operacion determinada con los valores a y b dependiendo del String operador
@author Joan
@date 12/05/2025
@param operador
@param a
@param b
@version 1.0
@deprecated usar el metodo operarRefactorizado
*/
```