



ENTORNOS DE DESARROLLO - HITO 5.1

SALMANE AABOUD

ÍNDICE:

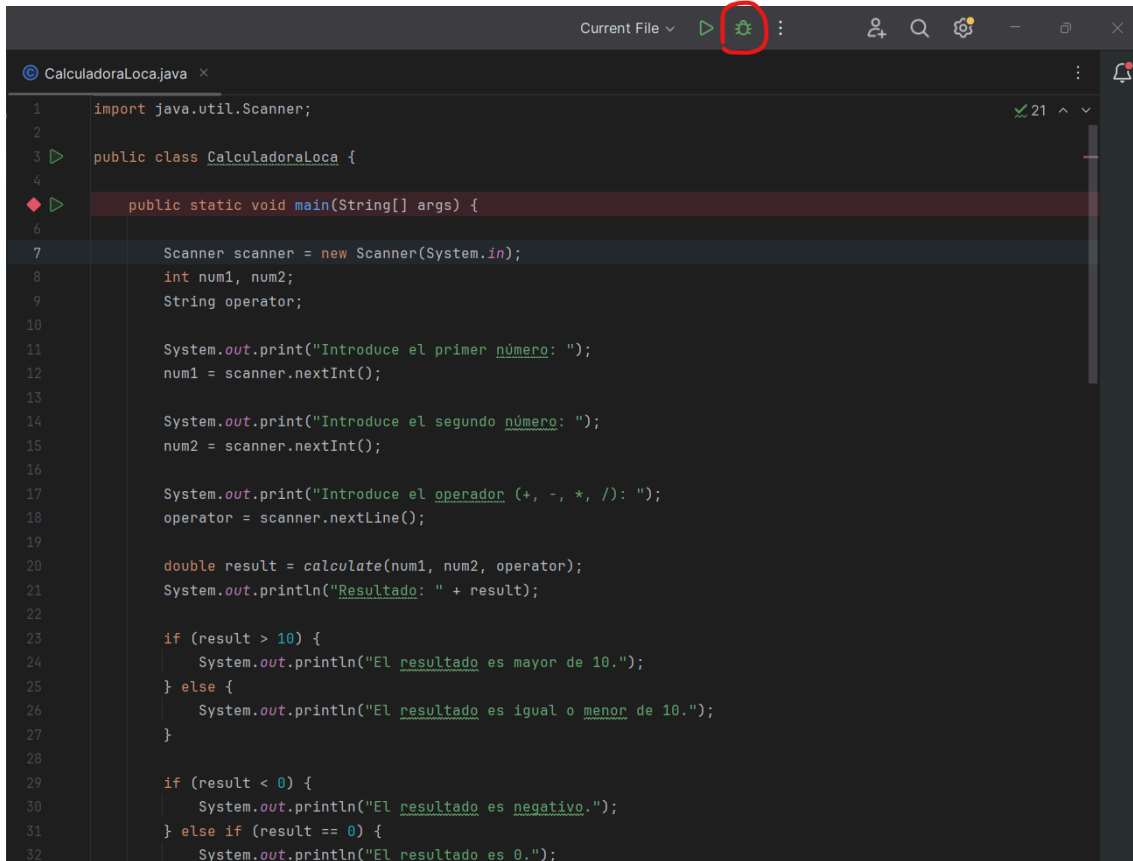
1. Introducción.....	3
2. Análisis del código original mediante el debugger.....	4
3. Corrección del código.....	9

1. Introducción

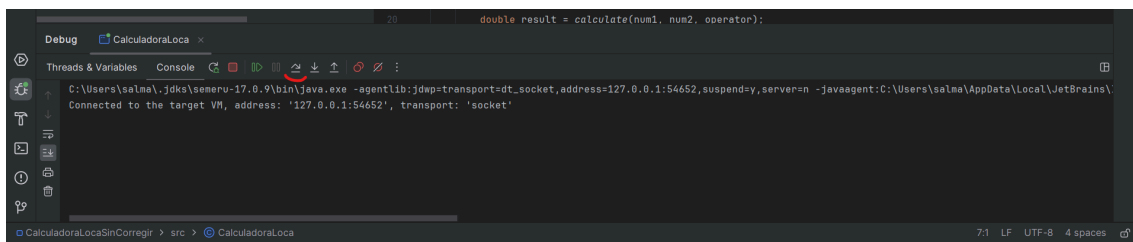
Se quiere trabajar el concepto de la depuración o *debugging* con el fin de conocer los beneficios que este nos proporciona a la hora de programar, ya que nos puede servir de gran ayuda para detectar los posibles fallos o *bugs* que contiene un código. Para este hito en concreto, se utilizará el depurador que trae IntelliJ por defecto para detectar los bugs que tiene un programa ideado como una calculadora.

2. Análisis del código original mediante el debugger

Este paso se basa en estudiar el código y sus métodos para entender el funcionamiento del programa. Una vez realizado esto, se ejecutará el código con el debugger de IntelliJ:

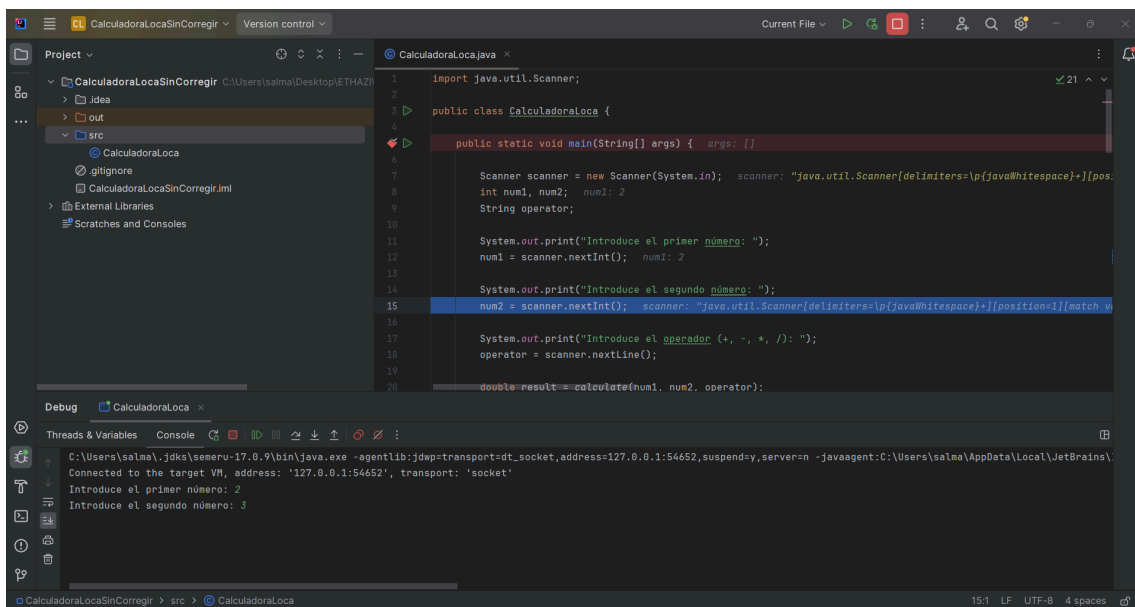
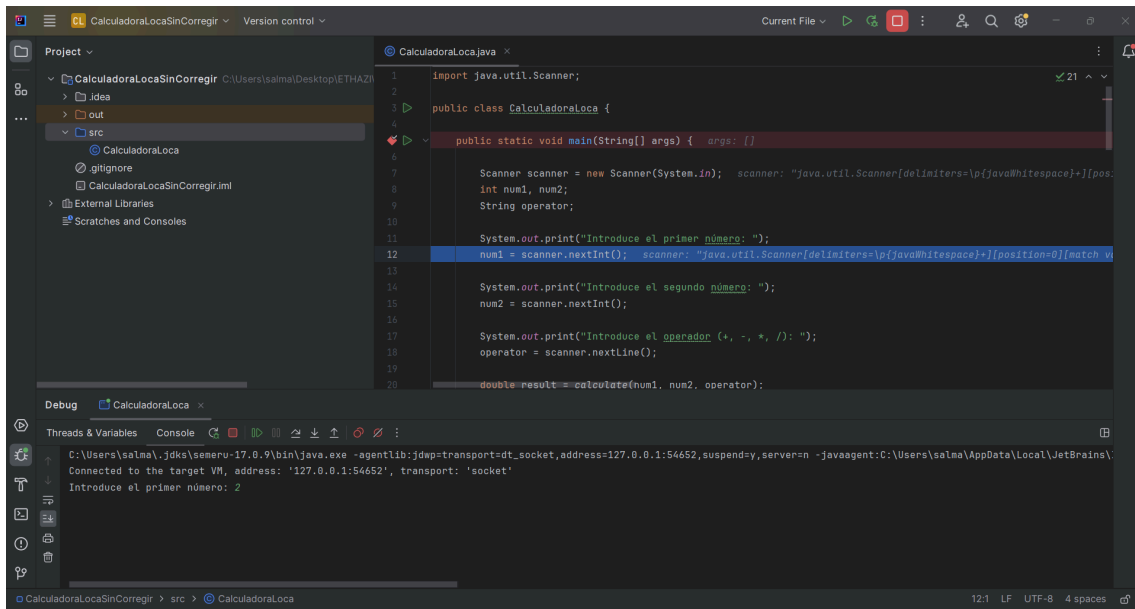


Abrimos la consola para visualizar los pasos que toma el programa. Para ir avanzando al siguiente paso que toma le damos clic al icono de “Step over”:

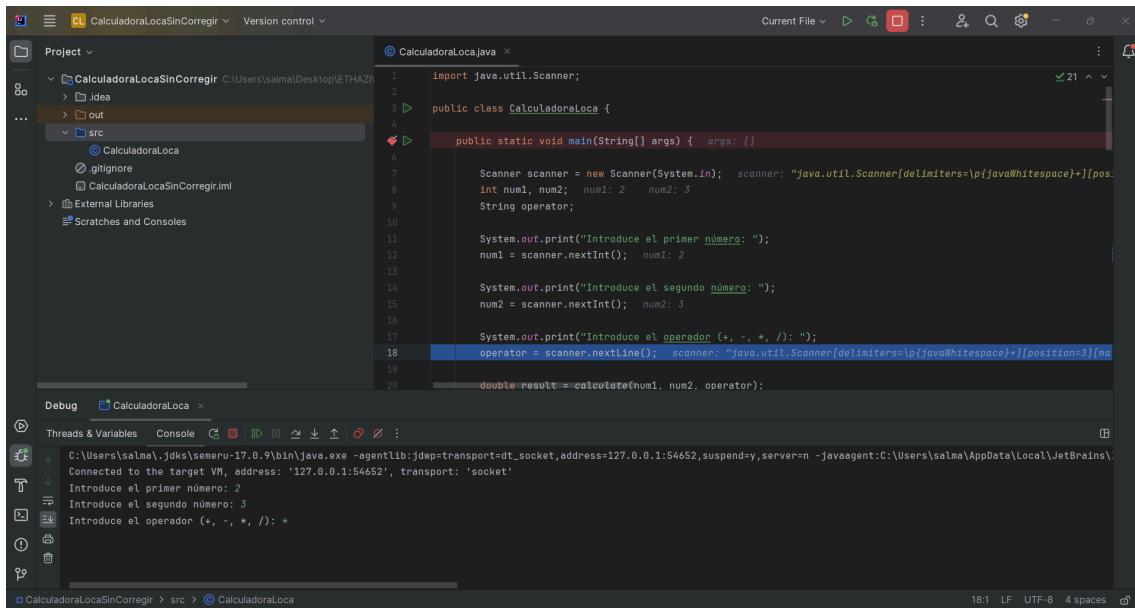


Insertamos los datos que nos pida que, en este caso, sería introducir los números con los que la calculadora va a operar:

- **NOTA:** En este paso, es imprescindible presionar Enter antes del *Step over* para poder pasar al siguiente punto del programa, ya que estamos operando con el Scanner de Java que obliga darle a la tecla para poder escanear los datos de entrada.



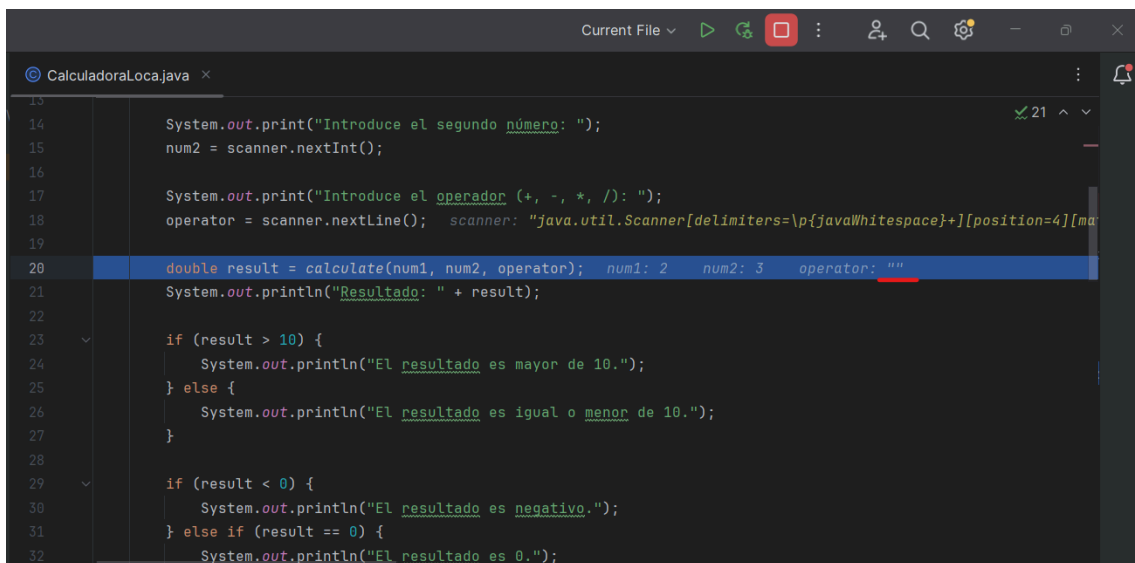
Seleccionamos el operador:



```
1 import java.util.Scanner;
2
3 public class CalculadoraLoca {
4
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         int num1, num2;  num1: 2  num2: 3
8         String operator;
9
10        System.out.print("Introduce el primer número: ");
11        num1 = scanner.nextInt();  num1: 2
12
13        System.out.print("Introduce el segundo número: ");
14        num2 = scanner.nextInt();  num2: 3
15
16        System.out.print("Introduce el operador (+, -, *, /): ");
17        operator = scanner.nextLine();  scanner: "java.util.Scanner[delimiters=\p{javaWhitespace}+][position=3][ma
18        double result = calculate(num1, num2, operator);
19    }
20 }
```

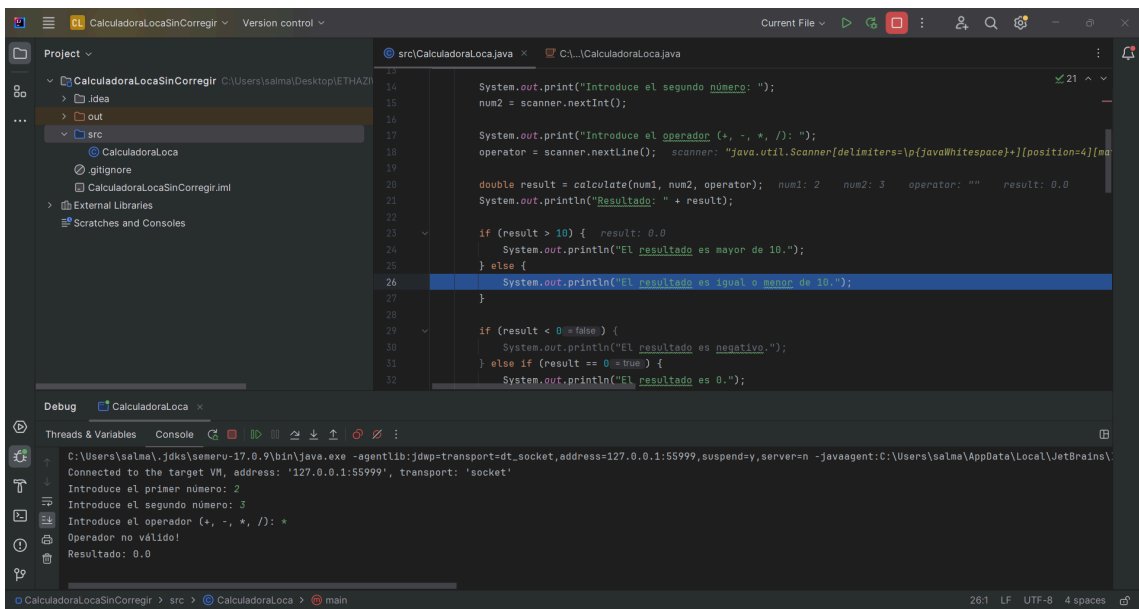
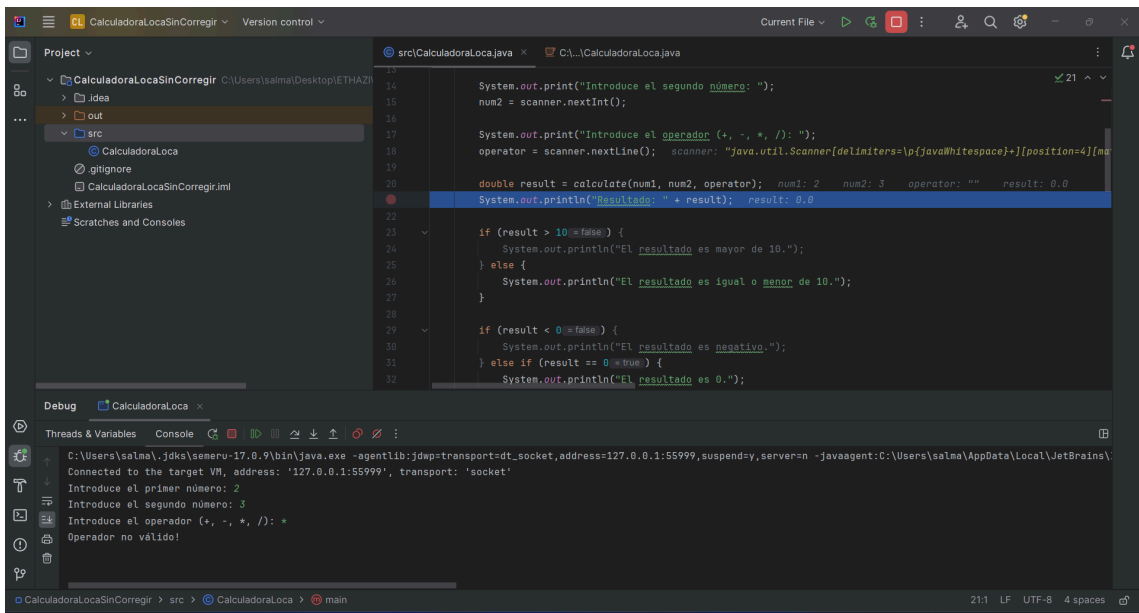
Debug Console:

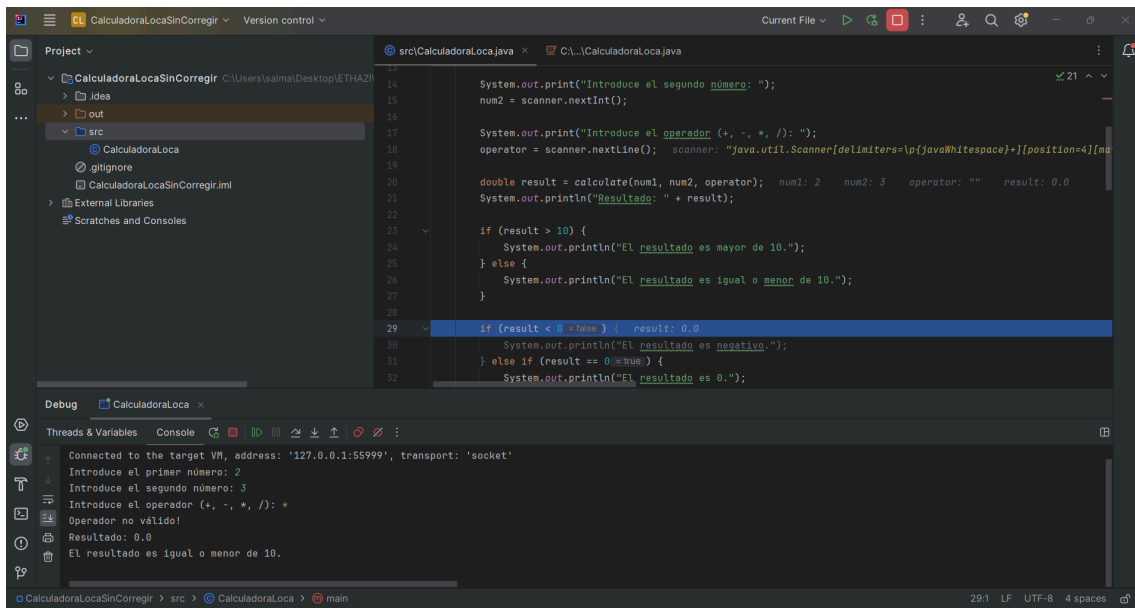
```
Introduce el primer número: 2
Introduce el segundo número: 3
Introduce el operador (+, -, *, /): *
```



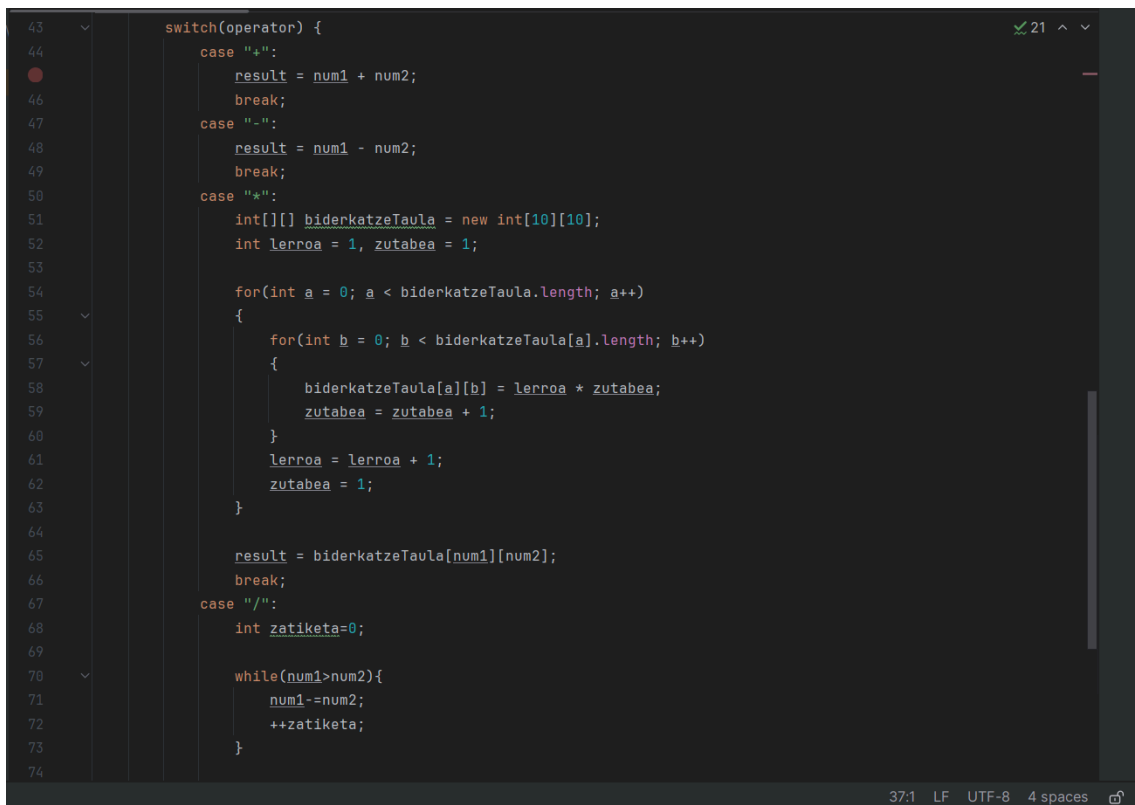
```
14 System.out.print("Introduce el segundo número: ");
15 num2 = scanner.nextInt();
16
17 System.out.print("Introduce el operador (+, -, *, /): ");
18 operator = scanner.nextLine();  scanner: "java.util.Scanner[delimiters=\p{javaWhitespace}+][position=4][ma
19
20 double result = calculate(num1, num2, operator);  num1: 2  num2: 3  operator: "*"
21 System.out.println("Resultado: " + result);
22
23 if (result > 10) {
24     System.out.println("El resultado es mayor de 10.");
25 } else {
26     System.out.println("El resultado es igual o menor de 10.");
27 }
28
29 if (result < 0) {
30     System.out.println("El resultado es negativo.");
31 } else if (result == 0) {
32     System.out.println("El resultado es 0.");
33 }
```

Vemos que no ha leído el operador que hemos escogido que, en este caso, era el de multiplicar “*”. Por lo tanto, este sería el primer error del código y el que debemos solucionar.





Como consecuencia, no va a entrar por ningún case del switch y pasará directamente al default:



default:

```
System.out.println("Operador no válido!");
```

3. Corrección del código

Como hemos mencionado anteriormente, el Scanner implementado que lee el operador no funciona correctamente. Para ello, se ha decidido reemplazar esta línea de código:

```
System.out.print("Introduce el segundo número: ");  
num2 = scanner.nextInt();
```

Por esta:

```
System.out.print("Introduce el operador (+, -, *, /): ");  
operator = scanner.next();
```

Se utiliza el next() en vez del nextLine() para leer únicamente el siguiente token, es decir, sólo un único carácter sin incluir el salto de línea.

Por otra parte, el código proporcionado no controla el hipotético caso de que el usuario divida entre 0 y, para ello, se he implementado el siguiente bloque dentro del caso del operador “/”:

```
case "/":  
    if (num2 != 0) {  
        result = (double) num1 / num2;  
    } else {  
        System.out.println("No se puede dividir por cero.");  
    }  
    break;
```

Además, como mejora, se quiere controlar la ejecución del siguiente fragmento:

```
double result = calculate(num1, num2, operator);
System.out.println("Resultado: " + result);

if (result > 10) {
    System.out.println("El resultado es mayor de 10.");
} else {
    System.out.println("El resultado es igual o menor de 10.");
}

if (result < 0) {
    System.out.println("El resultado es negativo.");
} else if (result == 0) {
    System.out.println("El resultado es 0.");
} else {
    System.out.println("El resultado es positivo.");
}
```

Como hemos mencionado previamente, el usuario podría pedir a la calculadora que divida un número por 0 y, para ello, se ha decidido implementar el código anterior. No obstante, nos conviene no ejecutar el código de arriba cuando suceda esto:

```

        double result = calculate(num1, num2, operator);

        checkResult(result);
    }

    public static void checkResult(double result){
        if(result!=0){
            System.out.println("Resultado: " + result);
            if (result > 10) {
                System.out.println("El resultado es mayor de 10.");
            } else {
                System.out.println("El resultado es igual o menor de 10.");
            }

            if (result < 0) {
                System.out.println("El resultado es negativo.");
            } else if (result == 0) {
                System.out.println("El resultado es 0.");
            } else {
                System.out.println("El resultado es positivo.");
            }
        }
    }
}

```

Hemos incluido el bloque en un método que tomará como parámetro el resultado que devuelva el otro método “calculate”. Comprobará que este no sea 0 para ejecute el fragmento mencionado.