

**ÍNDICE**:

[**1. Introducción 3**](#_heading=)

[**2. Análisis del código original mediante el debugger 4**](#_heading=)

[**3. Corrección del código 9**](#_heading=)

# 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.

# 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:

A screenshot of a computer program

Description automatically generated

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”:

A computer screen shot of a program

Description automatically generated

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.

A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Seleccionamos el operador:

A computer screen shot of a program

Description automatically generated

A screenshot of a computer program

Description automatically generated

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.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

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

A screen shot of a computer

Description automatically generated

A black background with white text

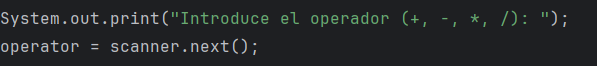
Description automatically generated

# 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:



Por esta:



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 “/”:

A computer screen shot of text

Description automatically generated

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

A computer screen with text

Description automatically generated

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:

A computer screen shot of code

Description automatically generated

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.