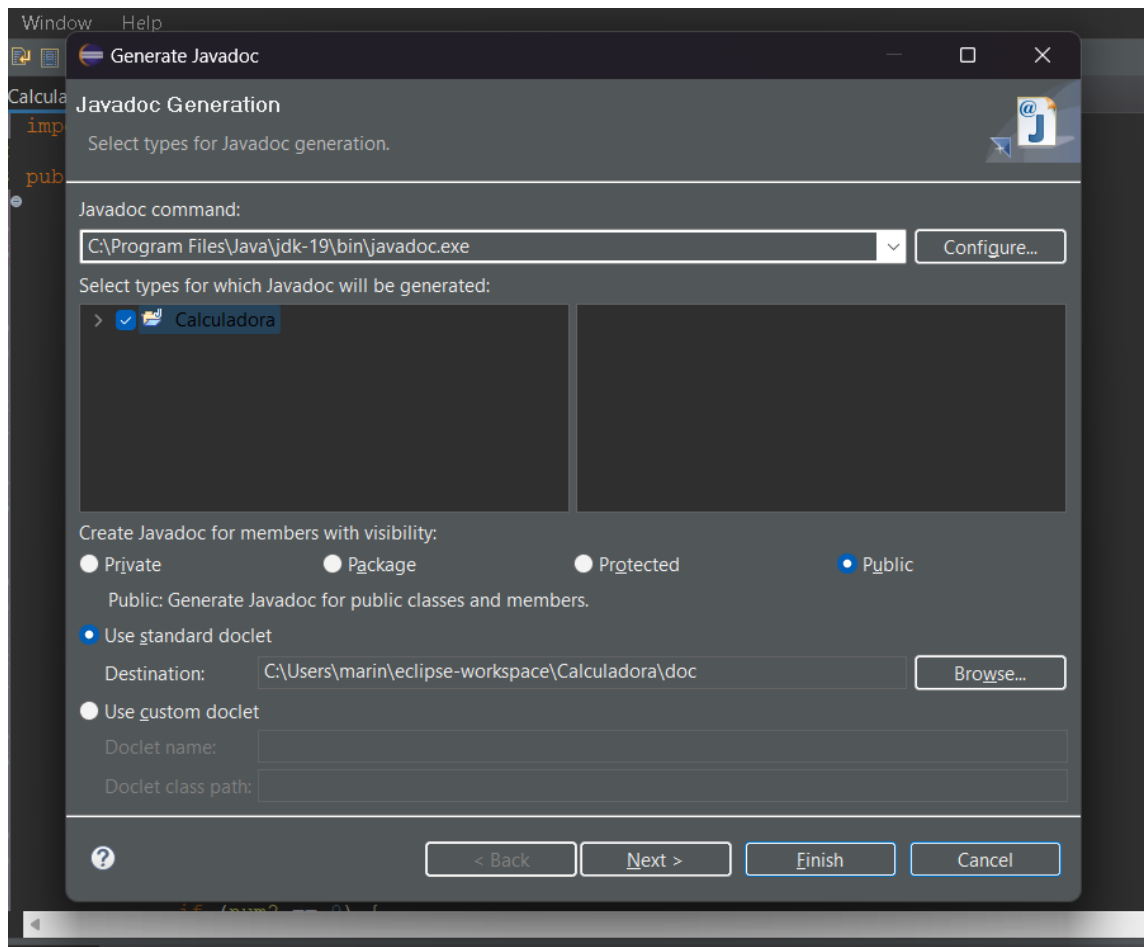
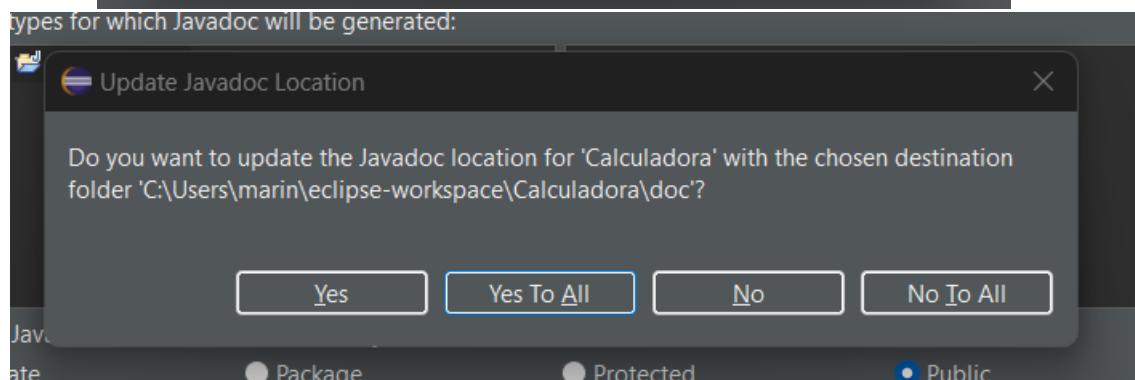
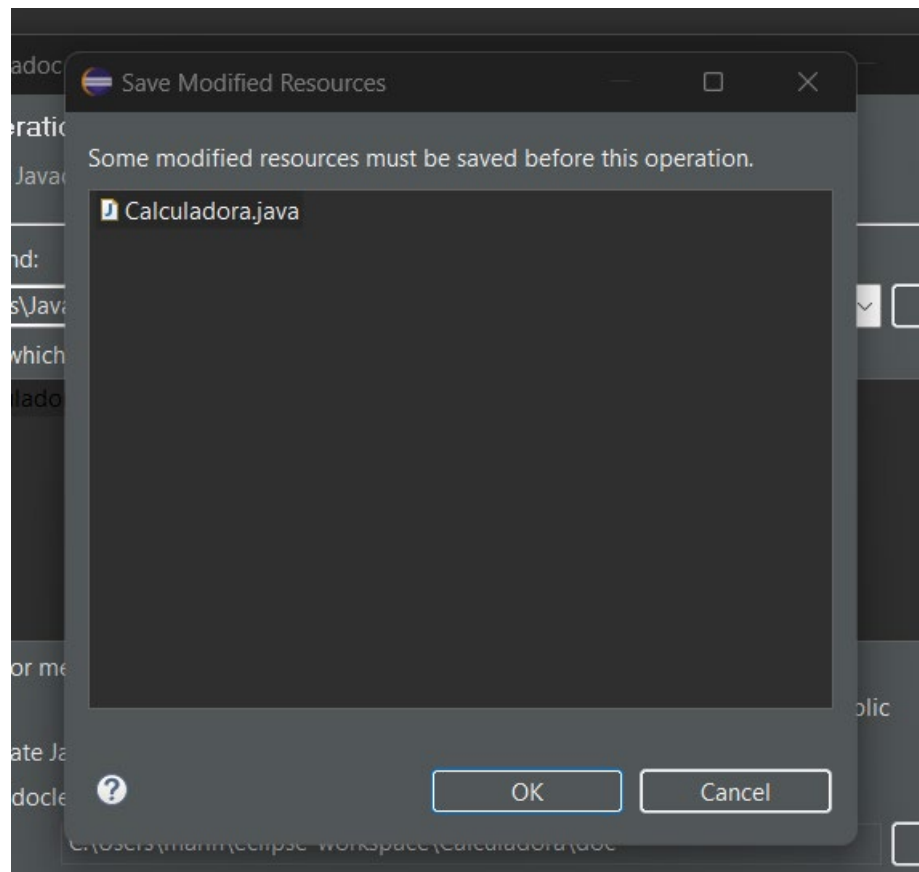


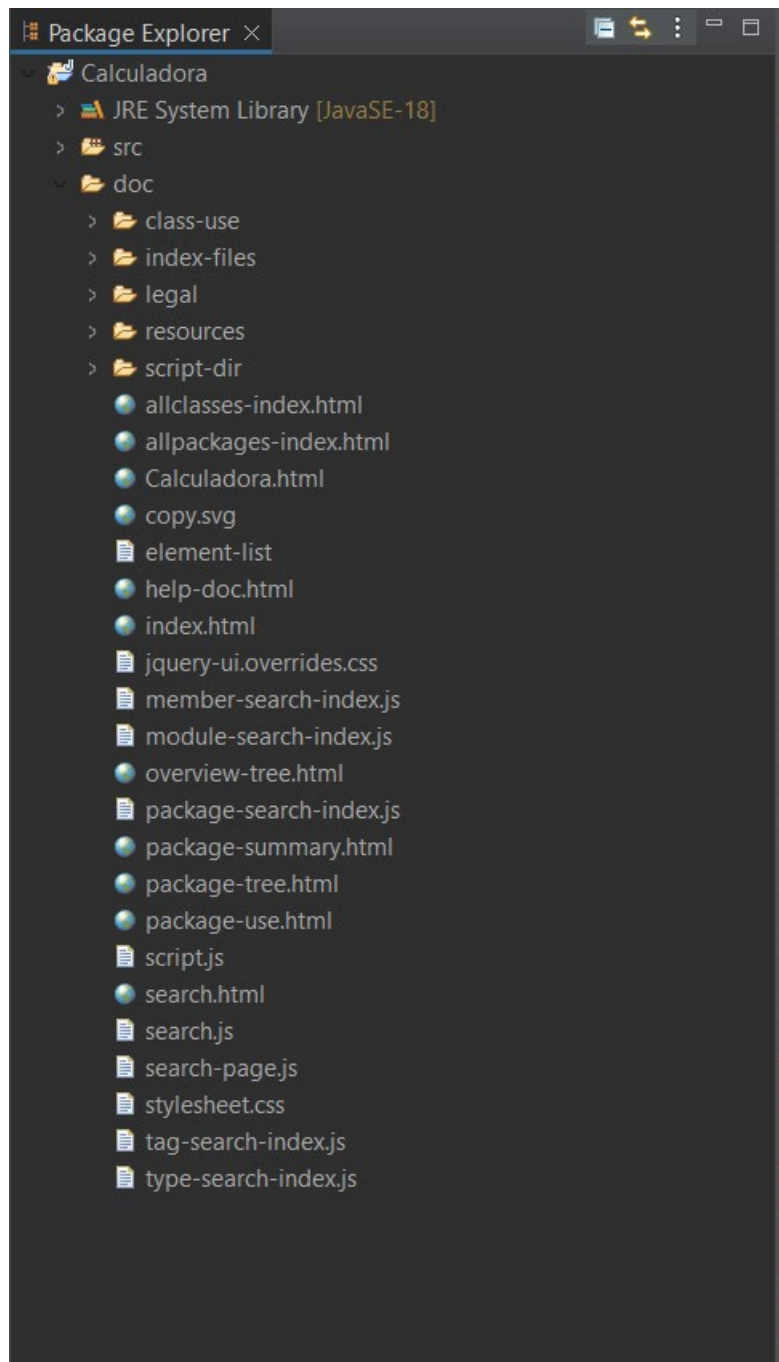
## Parte I – Javadoc

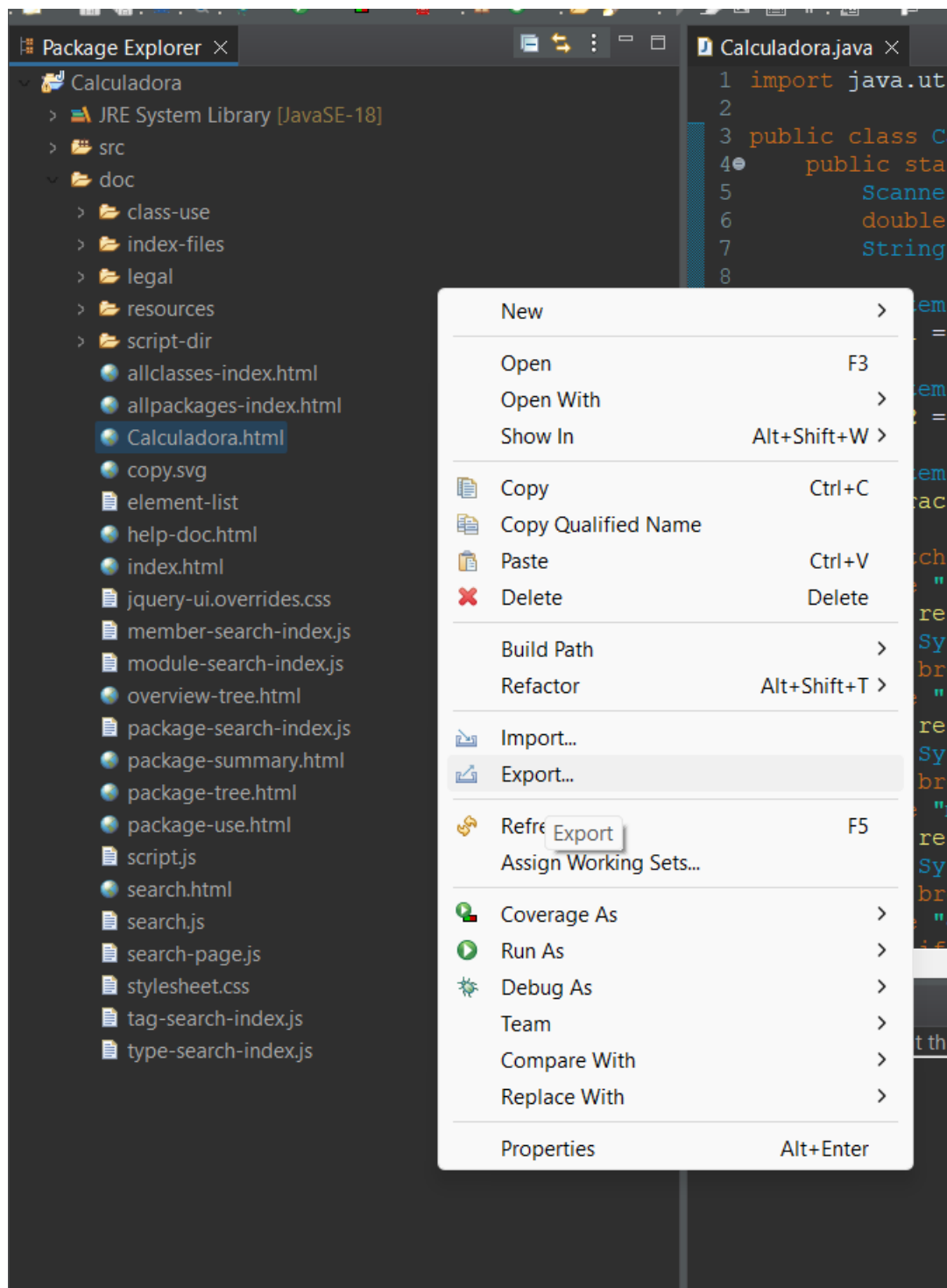
1. Elabora un manual de uso de javaDoc generado desde Eclipse con todas sus funcionalidades. Debes usar un ejemplo real en Java que contenga al menos una clase y alguna función.
2. Elabora un manual de uso de javaDoc generado desde IntelliJ con todas sus funcionalidades.

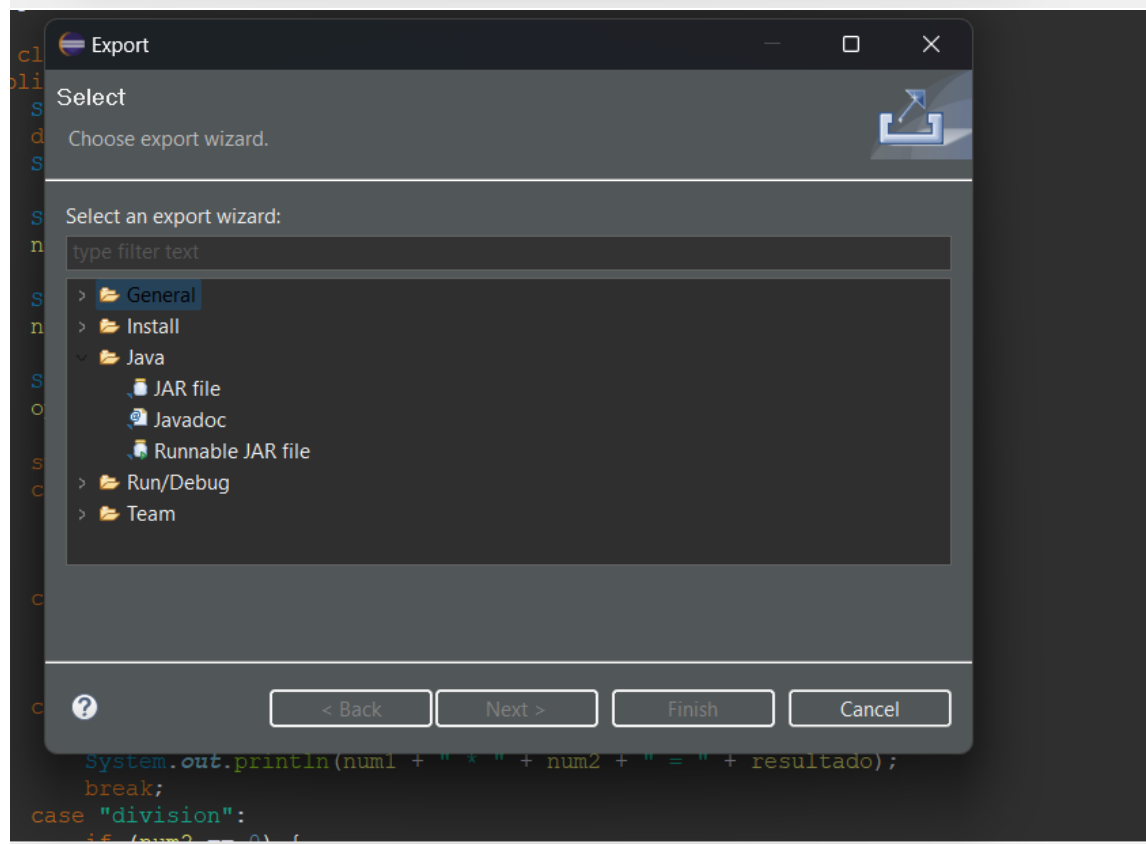
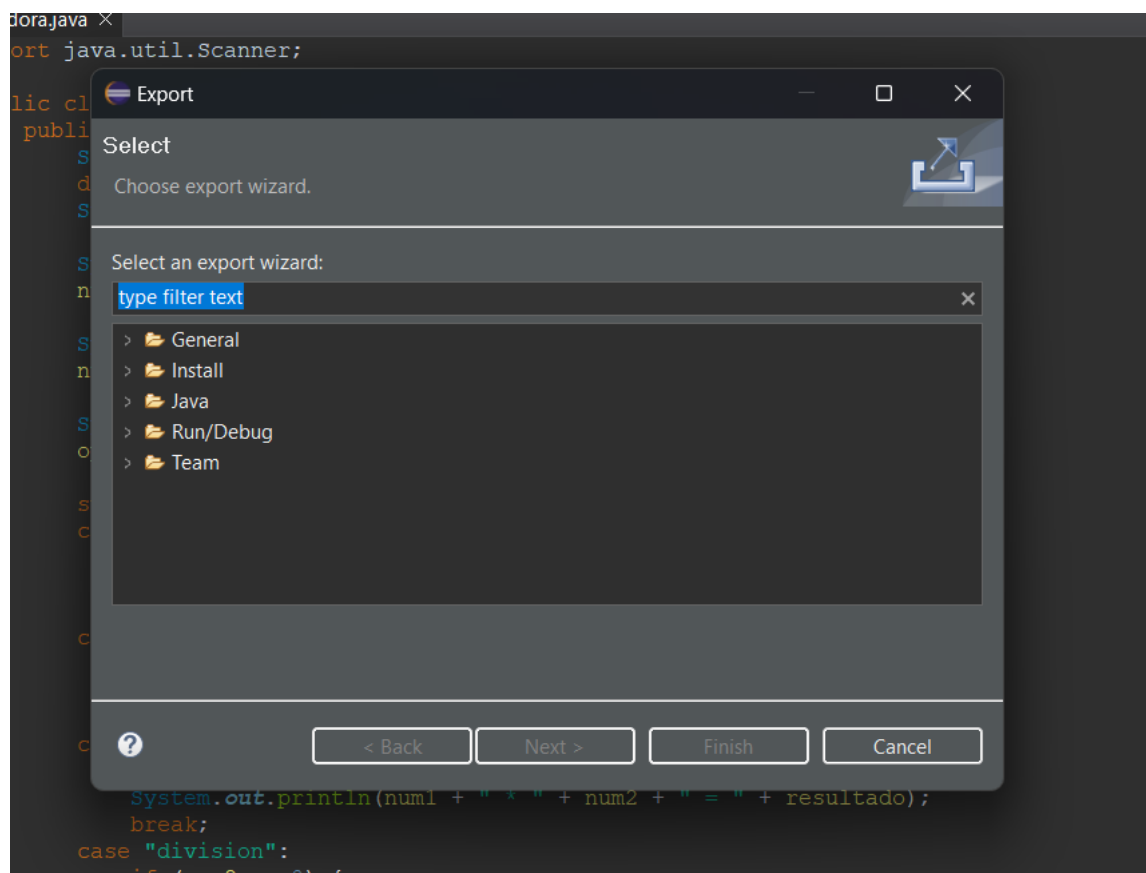
### MANUAL

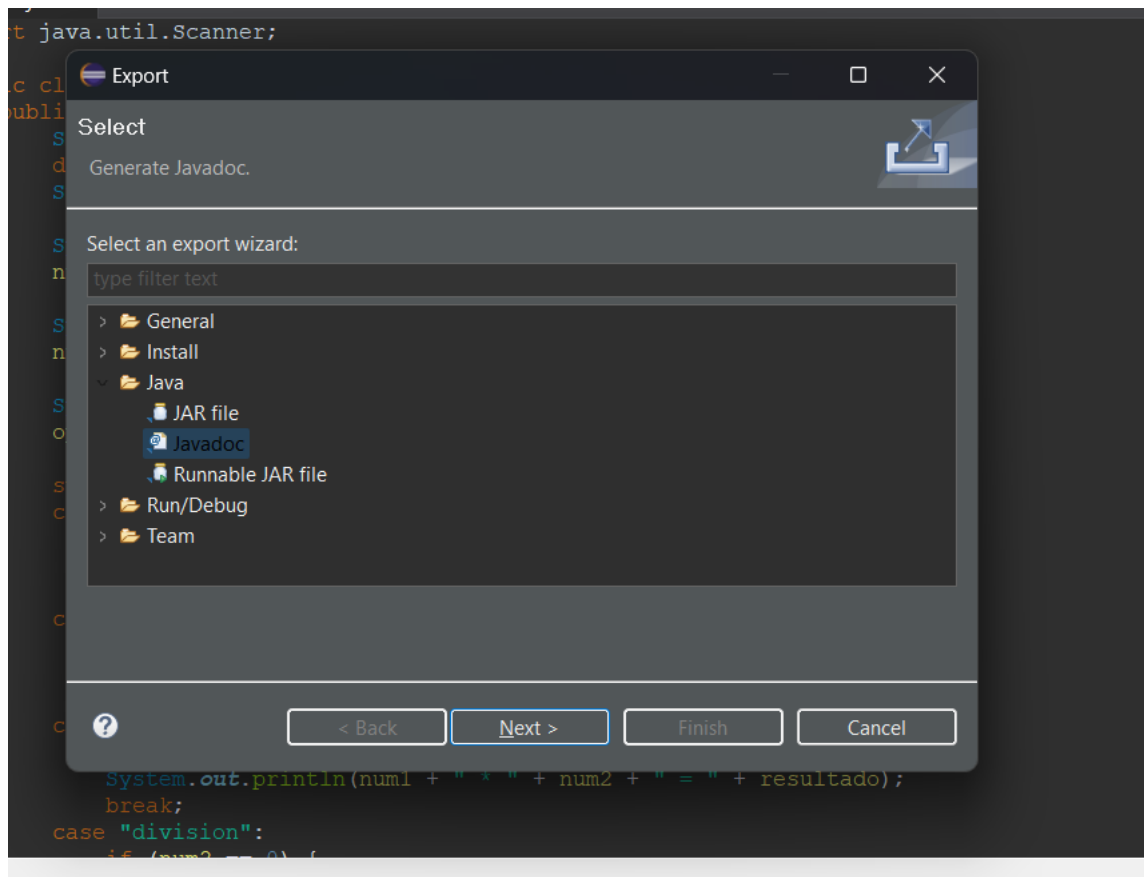


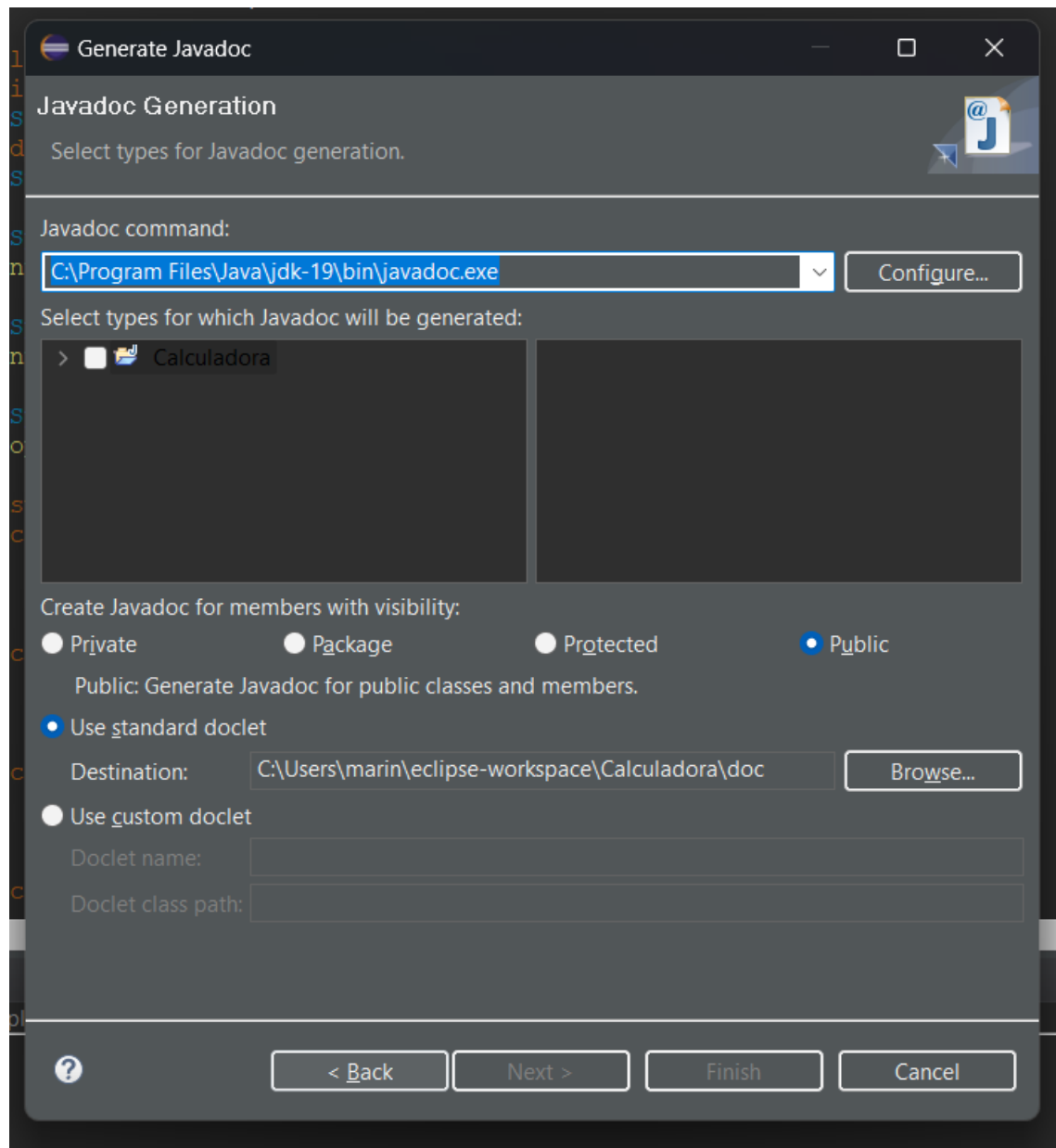








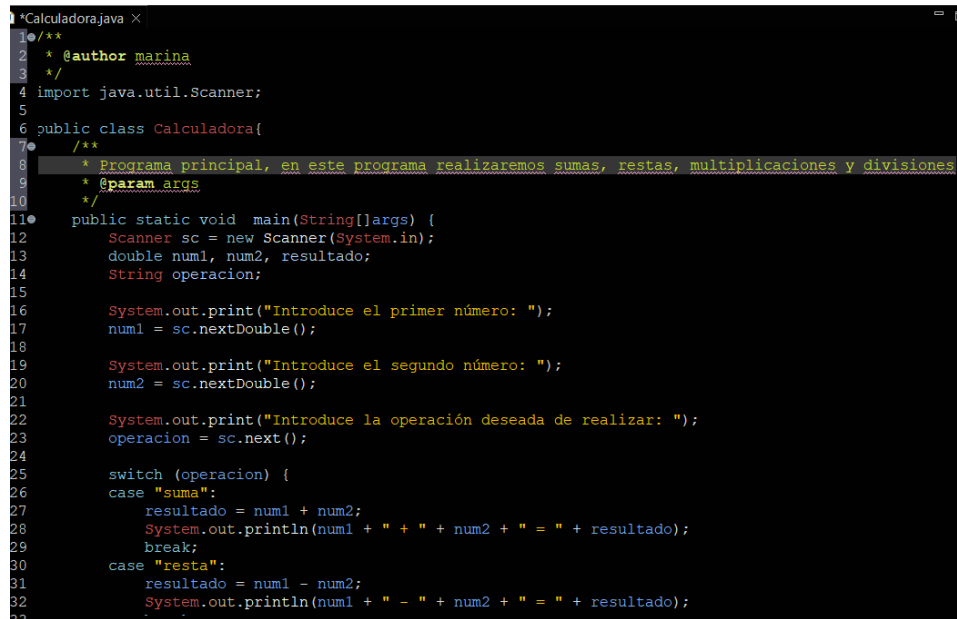




Para crear el Javadoc lo primero que debemos tener es el código del programa del que realizaremos el Javadoc. En el código empezaremos a poner los comentarios que corresponden al Javadoc su sintaxis es la siguiente.

```
/**  
 *  
 */
```

Cada párrafo nuevo que queramos poner empezara por un \*



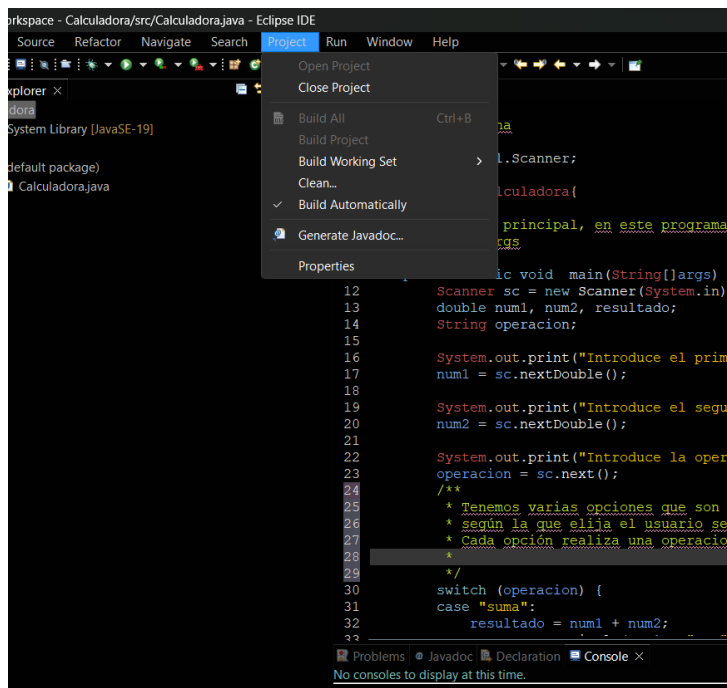
```
1  *Calculadora.java x  
2  /**  
3   * @author marina  
4   */  
5  
6  import java.util.Scanner;  
7  
8  public class Calculadora{  
9  
10     /**  
11      * Programa principal, en este programa realizaremos sumas, restas, multiplicaciones y divisiones  
12      * @param args  
13      */  
14     public static void main(String[] args) {  
15         Scanner sc = new Scanner(System.in);  
16         double num1, num2, resultado;  
17         String operacion;  
18  
19         System.out.print("Introduce el primer número: ");  
20         num1 = sc.nextDouble();  
21  
22         System.out.print("Introduce el segundo número: ");  
23         num2 = sc.nextDouble();  
24  
25         System.out.print("Introduce la operación deseada de realizar: ");  
26         operacion = sc.next();  
27  
28         switch (operacion) {  
29             case "suma":  
30                 resultado = num1 + num2;  
31                 System.out.println(num1 + " + " + num2 + " = " + resultado);  
32                 break;  
33             case "resta":  
34                 resultado = num1 - num2;  
35                 System.out.println(num1 + " - " + num2 + " = " + resultado);  
36                 break;  
37             default:  
38                 System.out.println("Operación no válida");  
39                 break;  
40         }  
41     }  
42 }  
43
```

Como observamos el primer comentario que comienza antes que el código del programa contiene algo específico que es el @author.

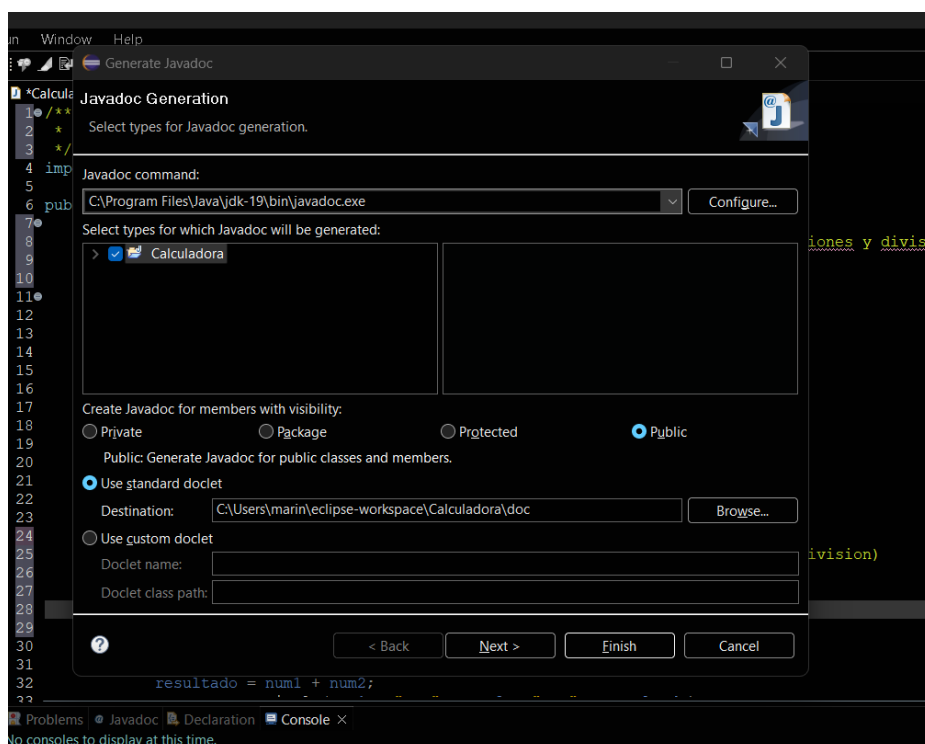
Mientras que el segundo comentario que observamos tiene una pequeña descripción de lo que va a realizar el programa principal y continuadamente vemos que tenemos una línea nueva en la que tenemos @param con esto lo que hacemos es especificar cuales son los parámetros que se le pasa a la función.

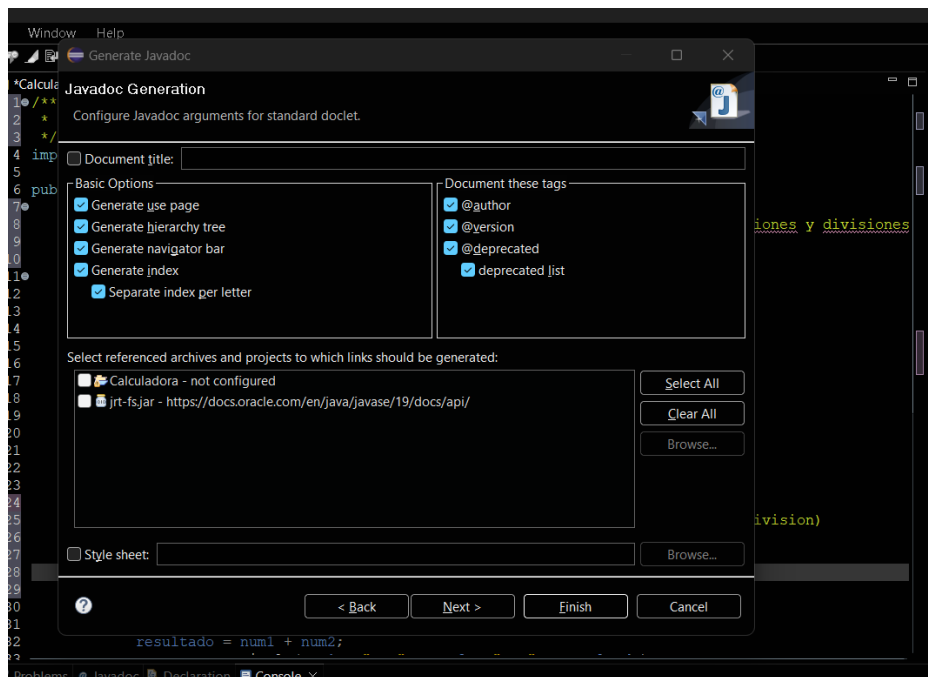
Una vez que tengamos nuestros comentarios colocados en el programa principal continuaremos con el siguiente paso a realizar, el cual es generar el Javadoc. Desde el IDE de Eclipse es tan fácil como entrar en el menú de Project el cual esta situado en la parte de arriba y veremos que en la penúltima posición nos aparece lo siguiente: Generate Javadoc...



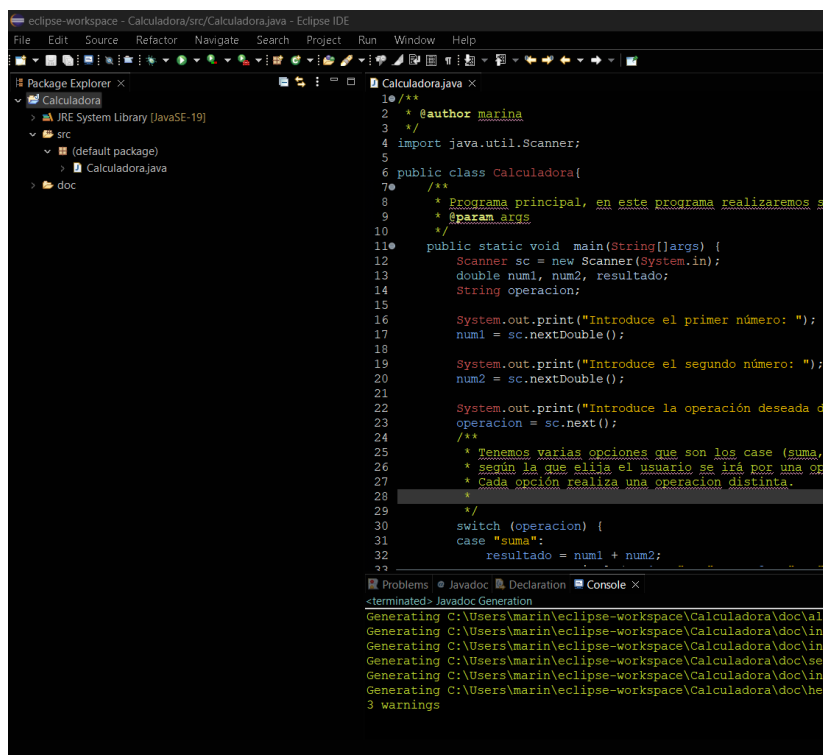


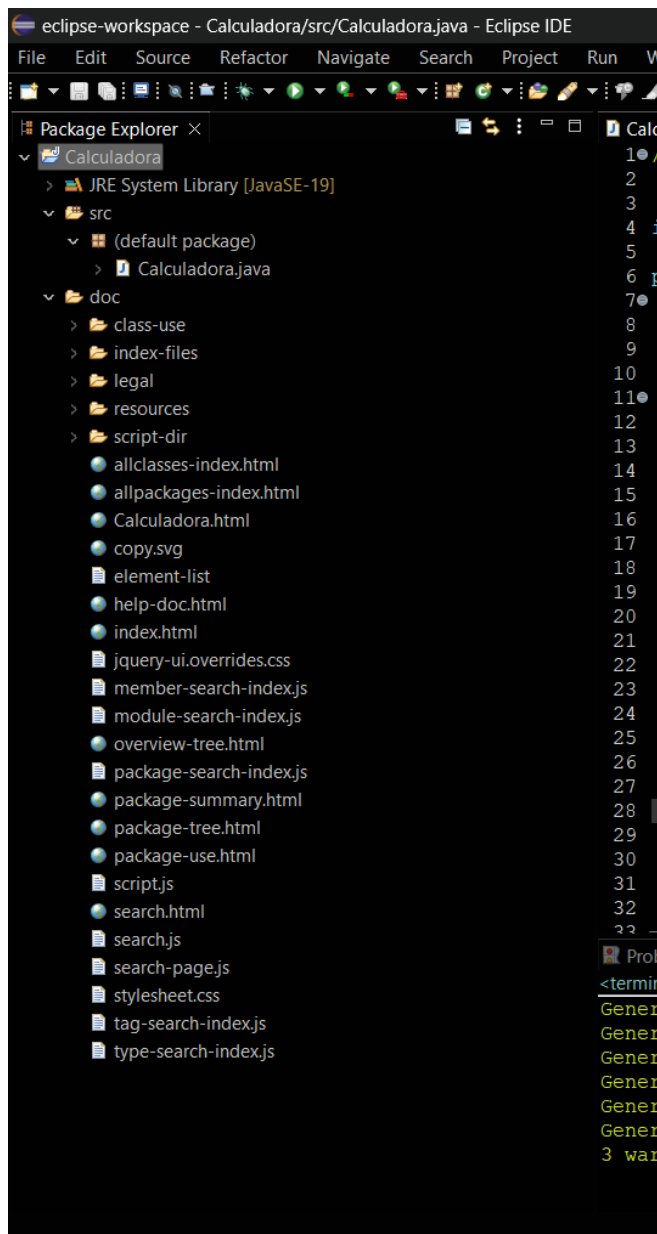
Seleccionaremos esta opción para generarlo y empezaremos a poner unos ajustes para poder generarlo, los cuales dejaremos los que nos vienen por defecto.





Una vez hayamos finalizado, observaremos como empieza a generarse el Javadoc y una vez finalizado encontraremos una nueva carpeta debajo de las carpetas de nuestro proyecto, esta carpeta tendrá el nombre de doc por defecto.





En esta carpeta encontraremos el archivo `index.html`, este es el archivo del Javadoc, el cual abriremos con el navegador para ver como ha quedado el documento que nos ha generado.

A continuación veremos como nos ha quedado el Javadoc de un programa algo básico.

En la primera página tendremos las clases de nuestro código.

PACKAGE CLASS USE TREE INDEX HELP	
PACKAGE: DESCRIPTION   RELATED PACKAGES   CLASSES AND INTERFACES	
Unnamed Package	
Classes	
Class	Description
Calculadora	

Pinchando en el nombre de la clase entraremos a su contenido. Podremos ver los constructores, métodos que nos encontremos en el código de esta clase.

PACKAGE CLASS USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD    DETAIL: FIELD | CONSTR | METHOD

Class Calculadora

java.lang.Object<sup>Ⓢ</sup>  
Calculadora

public class **Calculadora**  
extends Object<sup>Ⓢ</sup>

Constructor Summary

Constructors

Constructor	Description
calculadora()	

Method Summary

All Methods Static Methods Concrete Methods

Modifier and Type	Method	Description
static void	<b>main</b> (String <sup>Ⓢ</sup> [] args)	Programa principal, en este programa realizaremos sumas, restas, multiplicaciones y c

Methods inherited from class java.lang.Object<sup>Ⓢ</sup>

equals<sup>Ⓢ</sup>, getClass<sup>Ⓢ</sup>, hashCode<sup>Ⓢ</sup>, notify<sup>Ⓢ</sup>, notifyAll<sup>Ⓢ</sup>, toString<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>

Constructor Details

PACKAGECLASSUSE TREE INDEX HELP

SUMMARY NESTED FIELD CONSTR METHODDETAIL: FIELD CONSTR METHODSEARCH

All MethodsStatic MethodsConcrete Methods

Modifier and Type	Method	Description
static void	main(String[] args)	Programa principal, en este programa realizaremos sumas, restas, multiplicaciones y divisiones

Methods inherited from class java.lang.Object<sup>1</sup>  
equals<sup>1</sup>, getClass<sup>1</sup>, hashCode<sup>1</sup>, notify<sup>1</sup>, notifyAll<sup>1</sup>, toString<sup>1</sup>, wait<sup>1</sup>, wait<sup>1</sup>, wait<sup>1</sup>

Constructor Details

Calculadora

public Calculadora()

Method Details

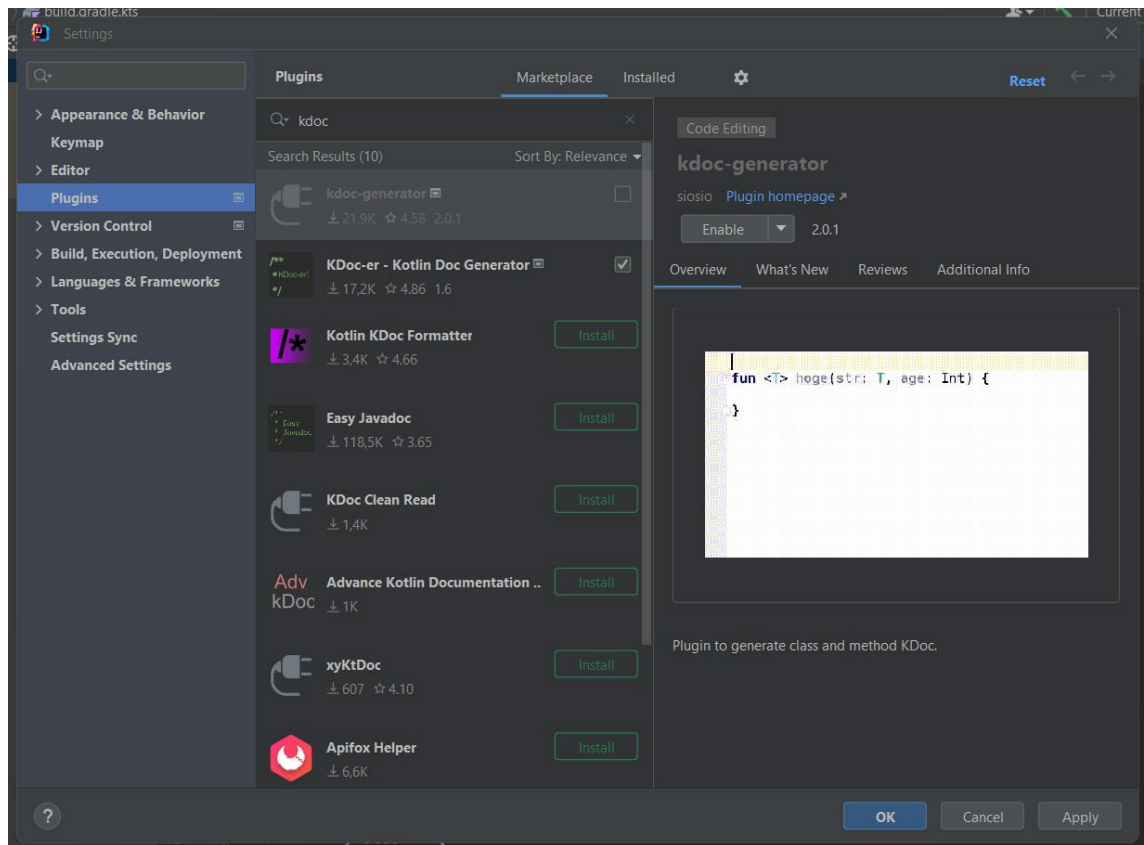
main

public static void main(String[] args)  
Programa principal, en este programa realizaremos sumas, restas, multiplicaciones y divisiones  
Parameters:  
args -

## Parte II – Kdoc

1. Elabora un manual de uso de KDoc generado desde IntelliJ con todas sus funcionalidades. Debes usar un ejemplo real de kotlin que contenga al menos una clase y alguna función.

### MANUAL



```
1 fun main(args: Array<String>) {  
2     plugins {  
3         id("org.jetbrains.dokka") version "1.7.20"  
4     }  
5 }
```

