



ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA



UNIVERSIDAD DE LAS FUERZAS ARMADAS “ESPE”

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

PRUEBAS DE SOFTWARE

PERÍODO MAYO/SEP /2023

INGENIERO:

ROLANDO REYES

NOMBRE:

ALISSON CLAVIJO

NRC: 9870

1. Introducción

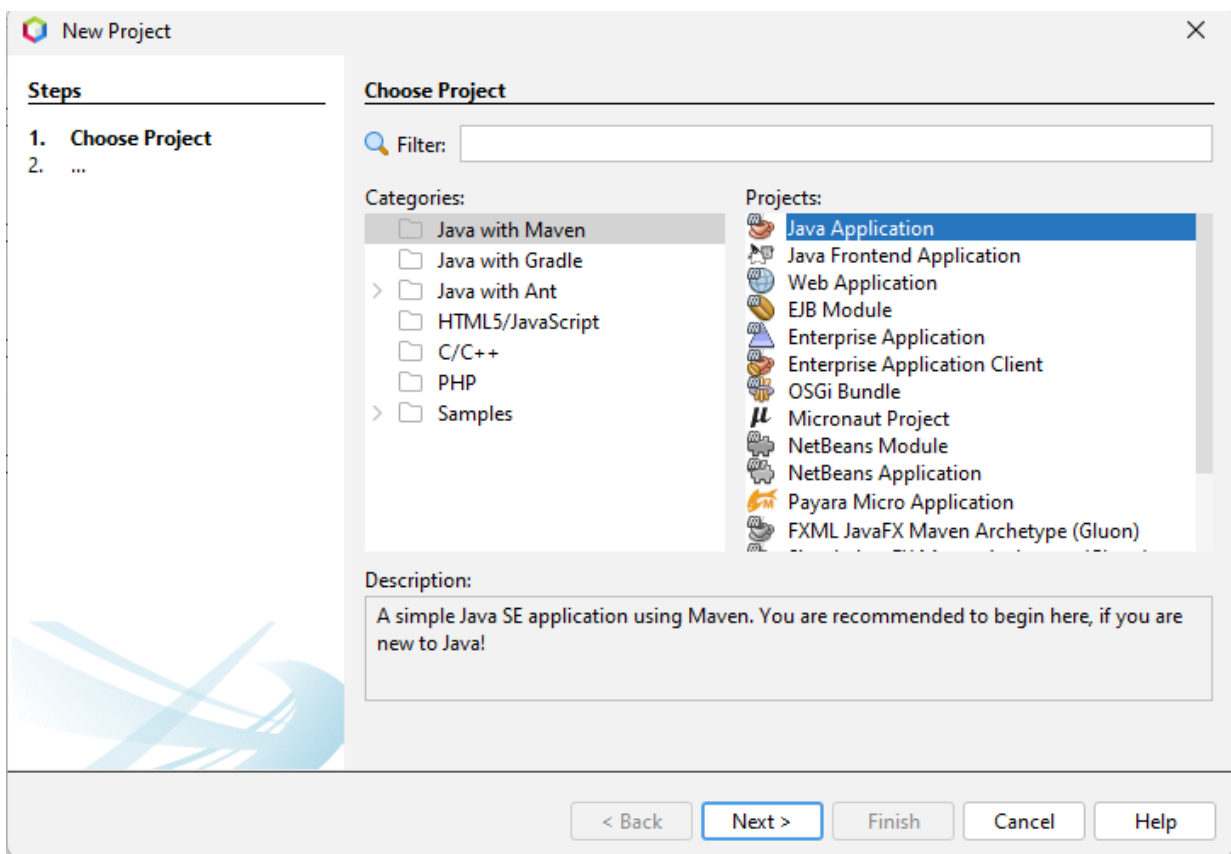
El objetivo de este informe es guiar y proporcionar instrucciones sobre cómo usar el plugin Java Code Coverage (JaCoCo) para el entorno de desarrollo Apache NetBeans 17 y probar la cobertura de código utilizando la herramienta de pruebas unitarias JUnit. El enfoque se centrará en probar el código proporcionado, que consiste en las clases `ArithmeticOperations.java` y `ArithmeticOperationsTest.java`, con el objetivo de lograr una cobertura del 100%.

2. Objetivos

- Descargar el plugin JaCoCo.
- Probar la cobertura del código.
- Generar un informe de cobertura.

3. Desarrollo

Primero se crea un nuevo proyecto



New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package: (Optional)

< Back Next > **Finish** Cancel Help

Creamos una Clase

New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name:

Project:

Location: ▼

Package: ▼

Created File:

Superclass:

Interfaces:

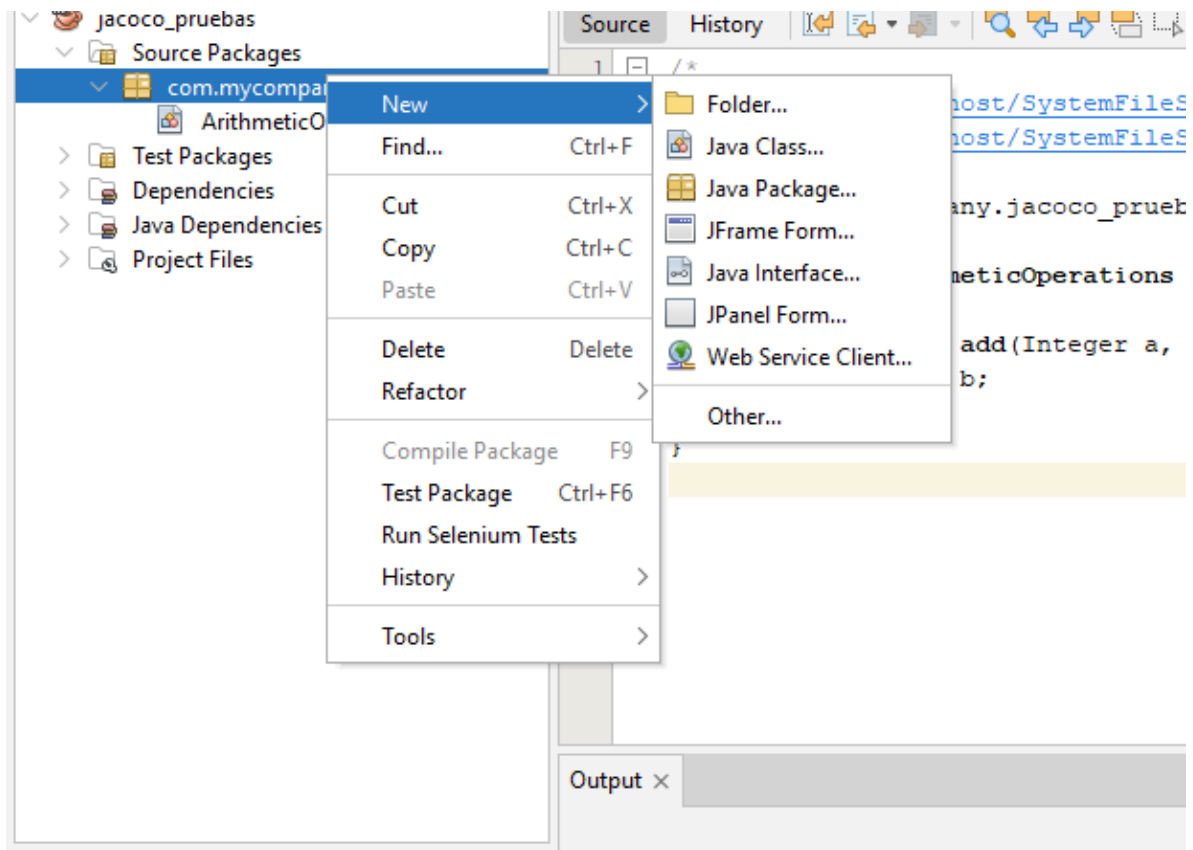
< Back Next > **Finish** Cancel Help

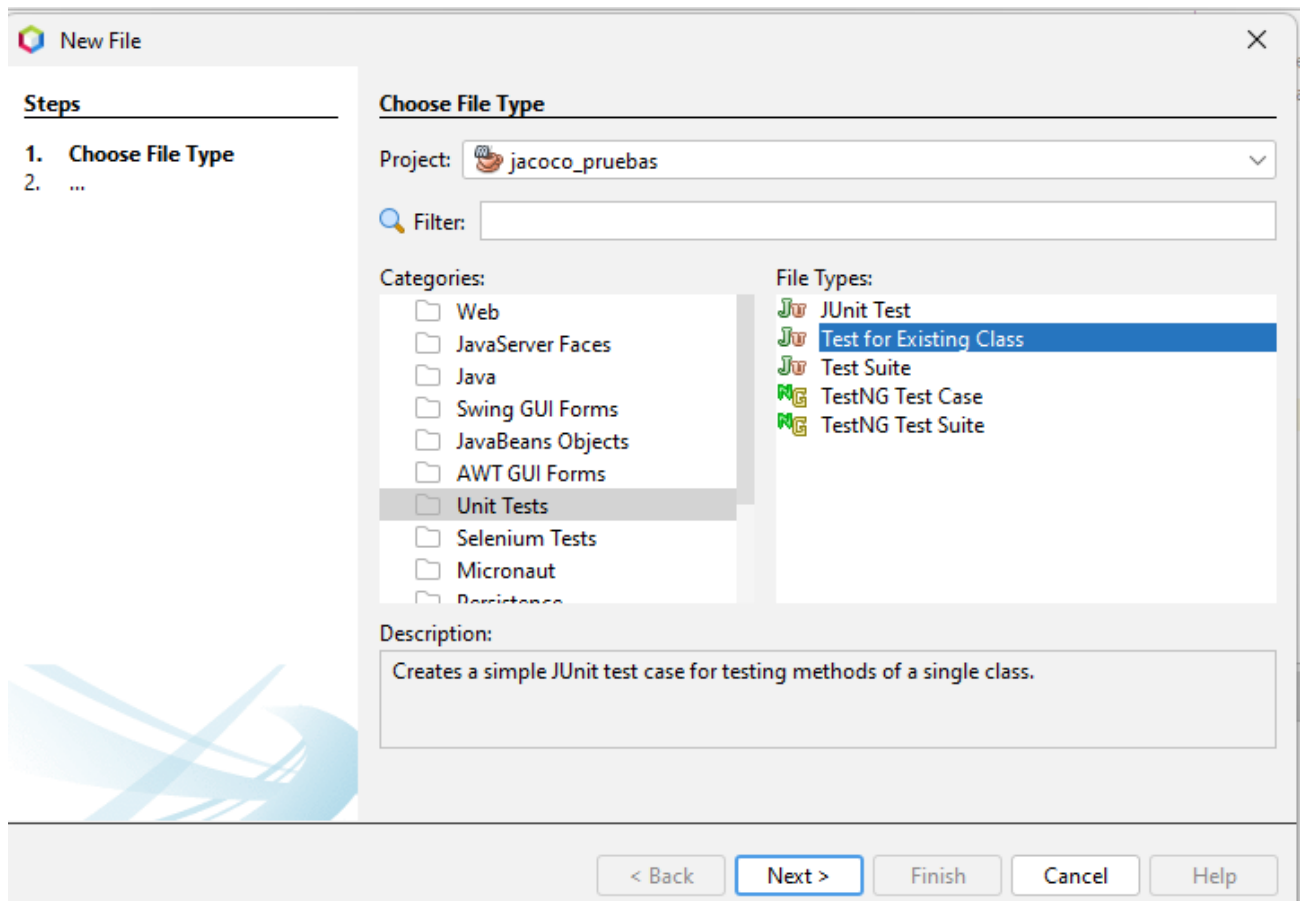
Copiamos el código



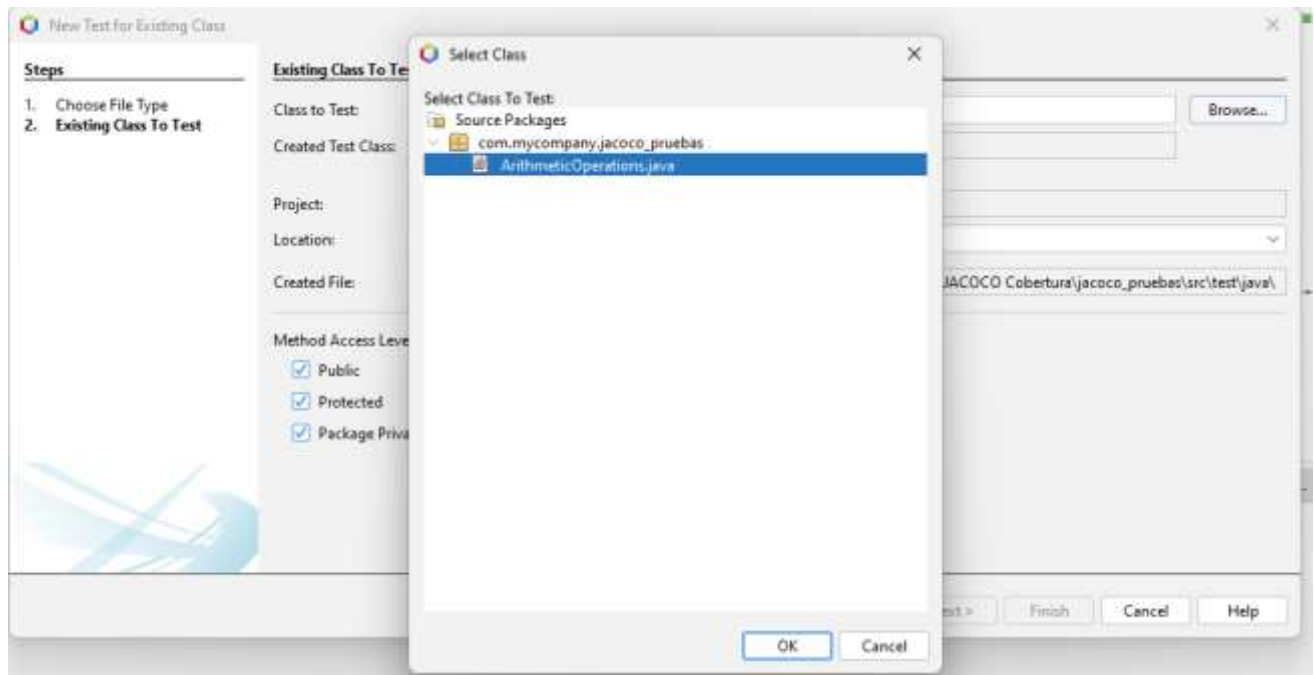
```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edi
4   */
5  package com.mycompany.jacoco_pruebas;
6
7  public class ArithmeticOperations {
8
9      public Integer add(Integer a, Integer b) {
10         return a + b;
11     }
12 }
13
```

Creamos el Unit Test





Seleccionamos la clase ya creada y que vamos a probar



New Test for Existing Class

Steps

1. Choose File Type

2. Existing Class To Test

Existing Class To Test

Class to Test:

com.mycompany.jacoco_pruebas.ArithmeticOperations

Browse...

Created Test Class:

com.mycompany.jacoco_pruebas.ArithmeticOperationsTest

Project:

jacoco_pruebas

Location:

Test Packages

Created File:

undo Parcial\JACOCO Cobertura\jacoco_pruebas\src\test\java\com\mycompany\jacoco_pruebas\ArithmeticOperationsTest.java

Method Access Levels

Generated Code

Generated Comments

☒ Public

☒ Test Initializer

☒ Javadoc Comments

☒ Protected

☒ Test Finalizer

☒ Source Code Hints

☒ Package Private

☒ Test Class Initializer

☒ Test Class Finalizer

☒ Default Method Bodies

< Back

Next >

Finish

Cancel

Help

Copiamos el código para realizar el Test

ProjectsFilesServices

jacoco_pruebas

Source Packages

com.mycompany.jacoco_pruebas

ArithmeticOperations.java

Test Packages

com.mycompany.jacoco_pruebas

ArithmeticOperationsTest.java

Dependencies

Test Dependencies

Java Dependencies

Project Files

testAdd - Navigator

Members

ArithmeticOperationsTest

ArithmeticOperationsTest()

testAdd()

Start PageArithmeticOperations.javaArithmeticOperationsTest.java

Source

History

import org.junit.jupiter.api.BeforeAll;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class ArithmeticOperationsTest {

@Test

public void testAdd()

{

ArithmeticOperations operations = new ArithmeticOperations();

Integer actual = operations.add(2, 6);

Integer expected = 8;

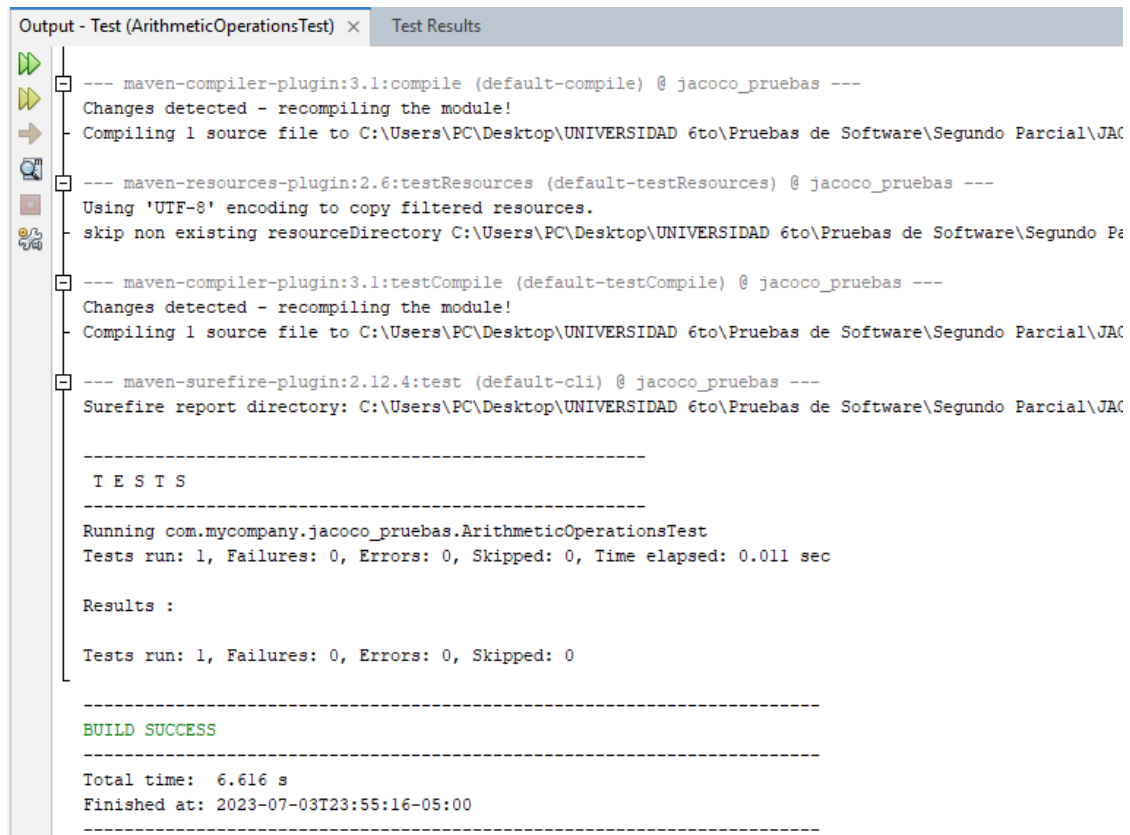
assertEquals(expected, actual);

}

}

Output

4. Resultados



```
--- maven-compiler-plugin:3.1:compile (default-compile) @ jacoco_pruebas ---
Changes detected - recompiling the module!
Compiling 1 source file to C:\Users\PC\Desktop\UNIVERSIDAD 6to\Pruebas de Software\Segundo Parcial\JAC

--- maven-resources-plugin:2.6:testResources (default-testResources) @ jacoco_pruebas ---
Using 'UTF-8' encoding to copy filtered resources.
skip non existing resourceDirectory C:\Users\PC\Desktop\UNIVERSIDAD 6to\Pruebas de Software\Segundo Pa

--- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ jacoco_pruebas ---
Changes detected - recompiling the module!
Compiling 1 source file to C:\Users\PC\Desktop\UNIVERSIDAD 6to\Pruebas de Software\Segundo Parcial\JAC

--- maven-surefire-plugin:2.12.4:test (default-cli) @ jacoco_pruebas ---
Surefire report directory: C:\Users\PC\Desktop\UNIVERSIDAD 6to\Pruebas de Software\Segundo Parcial\JAC

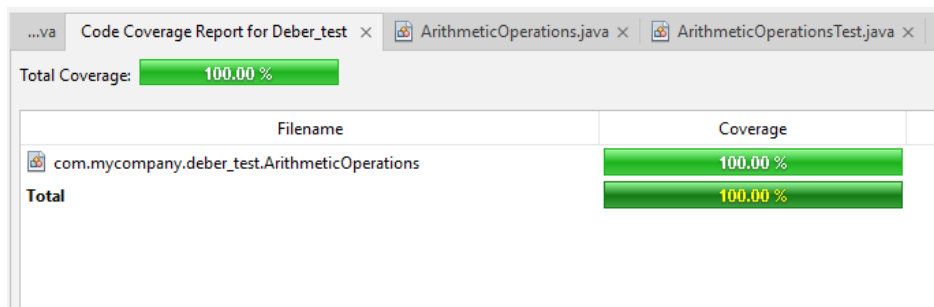
-----
T E S T S
-----
Running com.mycompany.jacoco_pruebas.ArithmeticOperationsTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.011 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 6.616 s
Finished at: 2023-07-03T23:55:16-05:00
-----
```

❖ 100% de cobertura



Filename	Coverage
com.mycompany.deber_test.ArithmeticOperations	100.00 %
Total	100.00 %



Filename	Coverage	Total	Not Executed
com.mycompany.deber_test.ArithmeticOperations	100.00 %	2	0
Total	100.00 %	2	0

Si modificamos el código la cobertura ya no será del 100%

The screenshot shows an IDE with two tabs: "ArithmeticOperations.java" and "ArithmeticOperationsTest.java". The "ArithmeticOperations.java" tab is active, displaying the following code:

```
5 package com.mycompany.deber_test;
6
7 /**
8  *
9  * @author PC
10  */
11 public class ArithmeticOperations {
12
13     public Integer add(Integer a, Integer b) {
14         if (a==b) {
15             return 0;
16         }
17         return a + b;
18     }
19 }
20
```

The code coverage report is displayed below the code editor. It shows a "Total Coverage: 75.00 %" with a green bar representing 75% and a red bar representing the remaining 25%. Below this, a table lists the files and their coverage:

Filename	Coverage
com.mycompany.deber_test.ArithmeticOperations	75.00 %
Total	75.00 %



5. Conclusiones y Recomendaciones

5.1 Conclusiones:

- El uso del plugin JaCoCo en Apache NetBeans 17 facilita la medición y análisis de la cobertura de código en pruebas unitarias, lo que ayuda a garantizar una mayor calidad del software.
- Las pruebas unitarias son esenciales para verificar la funcionalidad y detectar posibles errores en el código, y la cobertura de código nos permite identificar áreas no probadas o mal cubiertas.
- Al lograr una cobertura del 100%, se aumenta la confianza en la calidad del código, ya que se ha verificado exhaustivamente su funcionamiento.

5.2 Recomendaciones:

- Es recomendable utilizar el plugin JaCoCo en conjunto con Apache NetBeans 17 para medir la cobertura de código de manera efectiva y realizar mejoras en la calidad del software.
- Se debe garantizar que todas las clases y métodos importantes sean probados en las pruebas unitarias para obtener una cobertura adecuada.
- Es importante realizar pruebas exhaustivas y considerar diferentes escenarios para garantizar una cobertura completa del código.
- Se recomienda revisar regularmente los informes de cobertura generados por JaCoCo y realizar mejoras en las áreas con baja cobertura.

6. Bibliografía.

Diéguez, F. R. [@fernandorodriguezdieguez8577]. (2021, February 6). Jacoco: análisis de cobertura de código en Netbeans. Youtube. <https://www.youtube.com/watch?v=HQq9xei4gGE>