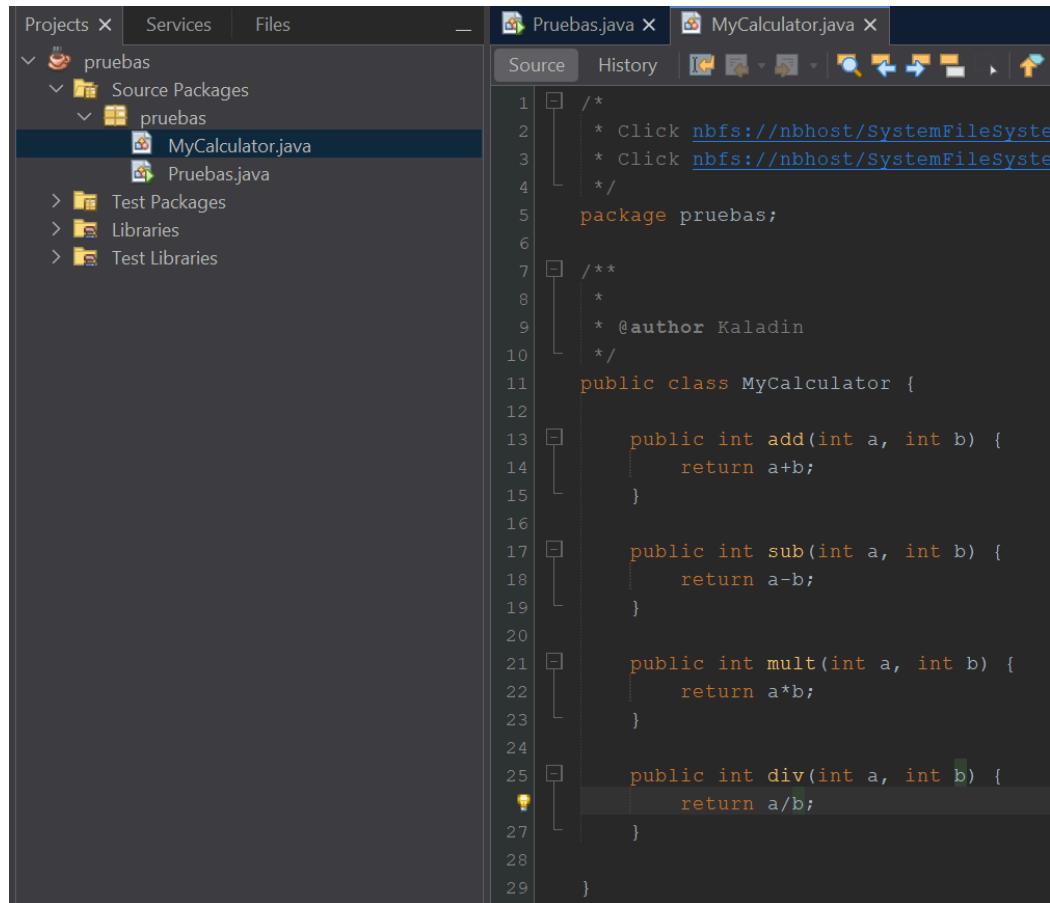


1.



The screenshot shows a Java code editor within an IDE. The left pane displays a project structure under the 'Projects' tab, showing a single package named 'pruebas' containing two files: 'MyCalculator.java' and 'Pruebas.java'. The right pane shows the source code for 'MyCalculator.java'. The code defines a class 'MyCalculator' with four methods: add, sub, mult, and div. The 'div' method contains a division operation 'a/b'. A yellow warning icon is visible next to the opening brace of the 'div' method's body.

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/1.5/MyCalculator.java to edit this file
3   * Click nbfs://nbhost/SystemFileSystem/1.5/Pruebas.java to edit this file
4   */
5  package pruebas;
6
7  /**
8   *
9   * @author Kaladin
10  */
11 public class MyCalculator {
12
13     public int add(int a, int b) {
14         return a+b;
15     }
16
17     public int sub(int a, int b) {
18         return a-b;
19     }
20
21     public int mult(int a, int b) {
22         return a*b;
23     }
24
25     public int div(int a, int b) {
26         return a/b;
27     }
28
29 }
```

## 2.

The screenshot shows a Java development environment with a project named "pruebas". The project structure on the left includes "Source Packages" (containing "pruebas" with files "MyCalculator.java" and "Pruebas.java"), "Test Packages" (containing a default package with "MyCalculatorTest.java"), "Libraries" (empty), and "Test Libraries" (containing JUnit 5.10.3 jars). The code editor on the right displays the "MyCalculatorTest.java" file. The code is a template for a JUnit 5 test class:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 * Click nbfs://nbhost/SystemFileSystem/Templates/UnitTests/JUnit5TestClass.j
 */
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

/**
 *
 * @author Kaladin
 */
public class MyCalculatorTest {

    public MyCalculatorTest() {
    }

    @Test
    void add() {
    }

    @BeforeAll
    public static void setUpClass() {
    }

    @AfterAll
    public static void tearDownClass() {
    }

    @BeforeEach
    public void setUp() {
    }

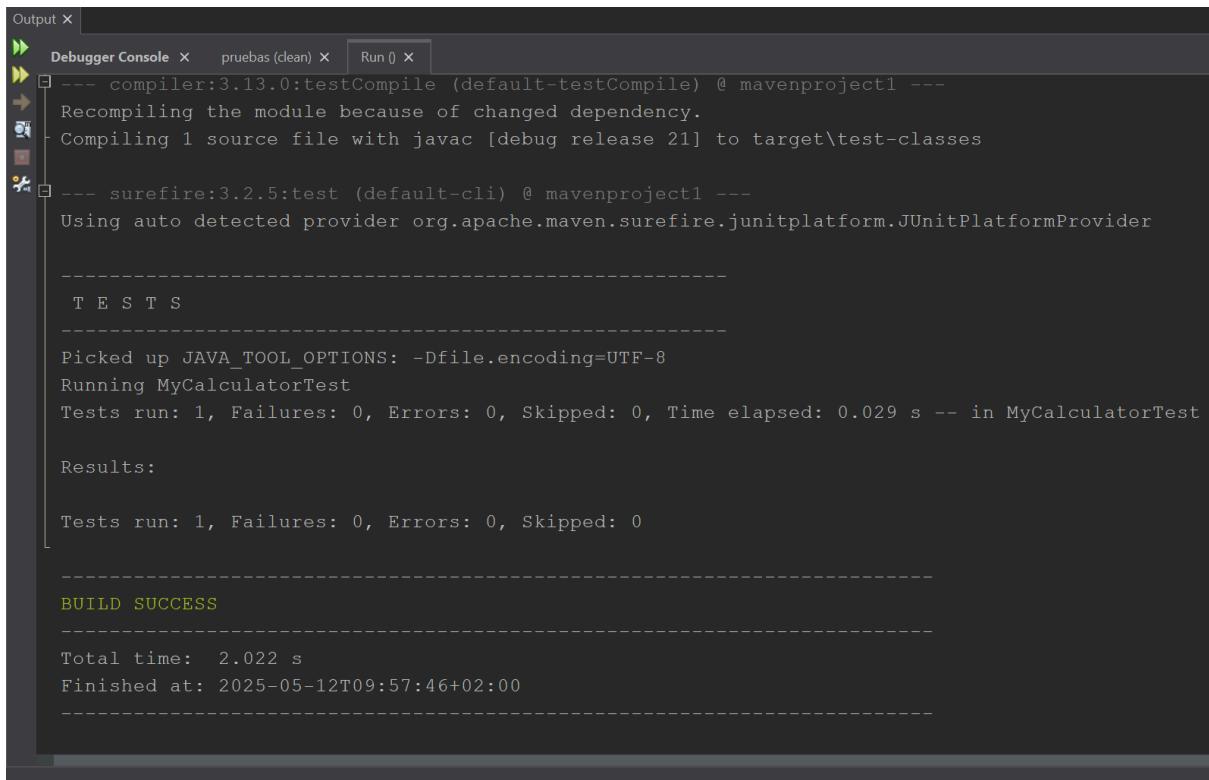
    @AfterEach
    public void tearDown() {
    }

    // TODO add test methods here.
    // The methods must be annotated with annotation @Test. For example:
    //
    // @Test
    // public void hello() {}
}
```

3.

```
public class MyCalculatorTest {  
    |  
    public MyCalculatorTest() {  
    }  
  
    @Test  
    void add() {  
        MyCalculator miCalculadora = new MyCalculator();  
  
        int resultadoEsperado = 10;  
        int resultadoActual = miCalculadora.add(5,5);  
  
        assertEquals(resultadoEsperado, resultadoActual);  
    }  
}
```

4.



The screenshot shows the IntelliJ IDEA interface with the 'Output' tab selected. The window displays the following log output:

```
Output ×  
▶ Debugger Console × pruebas (clean) × Run () ×  
--- compiler:3.13.0:testCompile (default-testCompile) @ mavenproject1 ---  
Recompiling the module because of changed dependency.  
Compiling 1 source file with javac [debug release 21] to target\test-classes  
--- surefire:3.2.5:test (default-cli) @ mavenproject1 ---  
Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider  
-----  
T E S T S  
-----  
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8  
Running MyCalculatorTest  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.029 s -- in MyCalculatorTest  
Results:  
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
-----  
BUILD SUCCESS  
-----  
Total time: 2.022 s  
Finished at: 2025-05-12T09:57:46+02:00  
-----
```

## 5.

The screenshot shows an IDE interface with two main panes. The top pane displays a Java test class named `MyCalculatorTest`. It contains three test methods: `sub()`, `mult()`, and `div()`. Each method creates a new instance of `MyCalculator`, performs a calculation, and then uses `assertEquals` to check if the result matches the expected value. The code is annotated with line numbers from 30 to 61. The bottom pane is the "Output" window, which shows the Maven build process and the execution of the tests. It starts with the compilation of the source code, followed by the execution of the test cases. The test results show that all four test methods passed successfully. The output concludes with a "BUILD SUCCESS" message and the total execution time.

```
30     assertEquals(resultadoEsperado, resultadoActual);
31 }
32
33 @Test
34 void sub() {
35     MyCalculator miCalculadora = new MyCalculator();
36
37     int resultadoEsperado = 1;
38     int resultadoActual = miCalculadora.sub(6, 5);
39
40     assertEquals(resultadoEsperado, resultadoActual);
41 }
42
43 @Test
44 void mult() {
45     MyCalculator miCalculadora = new MyCalculator();
46
47     int resultadoEsperado = 16;
48     int resultadoActual = miCalculadora.mult(4, 4);
49
50     assertEquals(resultadoEsperado, resultadoActual);
51 }
52
53 @Test
54 void div() {
55     MyCalculator miCalculadora = new MyCalculator();
56
57     int resultadoEsperado = 2;
58     int resultadoActual = miCalculadora.div(6, 3);
59
60     assertEquals(resultadoEsperado, resultadoActual);
61 }
```

Output X

```
pruebas (clean) × Test (MyCalculatorTest) ×
--- compiler:3.13.0:testCompile (default-testCompile) @ mavenproject1 ---
Recompiling the module because of changed source code.
Compiling 1 source file with javac [debug release 21] to target\test-classes

--- surefire:3.2.5:test (default-cli) @ mavenproject1 ---
Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider

-----
T E S T S
-----
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Running MyCalculatorTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.031 s -- in MyCalculatorTest

Results:

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 1.294 s
Finished at: 2025-05-12T10:06:19+02:00
-----
```

6.

7.

The screenshot shows an IDE interface with two main panes. The top pane displays a Java test class named MyCalculatorTest. The code includes setup and teardown methods for each test, and two test methods (add() and sub()) that perform arithmetic operations and assert the results. The bottom pane is the 'Output' window, which shows the execution of the tests. It prints messages from the test code indicating the setup and teardown of the calculator for each test case. It also shows the results of the four tests run, stating there were 4 tests, 0 failures, 0 errors, and 0 skipped tests, with a total time elapsed of 0.036 seconds. The output concludes with a 'BUILD SUCCESS' message and the total execution time and date.

```
24
25     @BeforeEach
26     void setUpMyCalculator() {
27         System.out.println("@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba");
28         miCalculadora = new MyCalculator();
29     }
30
31     @AfterEach
32     void tearDownMyCalculator() {
33         System.out.println("@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba");
34         miCalculadora = null;
35     }
36
37     @Test
38     void add() {
39
40         int resultadoEsperado = 10;
41         int resultadoActual = miCalculadora.add(5, 5);
42
43         assertEquals(resultadoEsperado, resultadoActual);
44     }
45
46     @Test
47     void sub() {
48
49         int resultadoEsperado = 1;
50         int resultadoActual = miCalculadora.sub(6, 5);
51
52         assertEquals(resultadoEsperado, resultadoActual);
53     }

```

```
Output ×
pruebas (clean) × Test (MyCalculatorTest) ×
T E S T S
-----
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Running MyCalculatorTest
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.036 s -- in MyCalculatorTest

Results:

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0

-----
BUILD SUCCESS
-----
Total time: 1.316 s
Finished at: 2025-05-12T10:14:05+02:00
-----
```

8.

The screenshot shows an IDE interface with two main panes. The top pane displays Java code for a unit test named `divByZero`. The code attempts to divide by zero and assert that the error message contains "División por cero". The bottom pane shows the test results: 1 test passed (80.00%) and 1 test failed (MyCalculatorTest.Failed). The failed test is labeled "divByZero Failed: expected: <true> but was: <false>". The output pane on the right shows repeated log entries for setup and teardown methods.

```
36
37     @Test
38     void divByZero() {
39         int resultadoActual = 0;
40         String mensajeEsperado = "";
41
42         try {
43             resultadoActual = miCalculadora.div(6, 0);
44         } catch (ArithmetricException e) {
45             mensajeEsperado = "División por cero";
46             assertTrue(mensajeEsperado.contains(e.getMessage())));
47         }
48     }
49 
```

Output Test Results x  
MyCalculatorTest.add x com.mycompany:mavenproject1:jar:1.0-SNAPSHOT (Unit) x  
Tests passed: 80.00 %  
1 tests passed, 1 test failed: (0.043 s)  
MyCalculatorTest Failed  
divByZero Failed: expected: <true> but was: <false>

@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba

9.

The screenshot shows an IDE interface with two main panes. The top pane displays Java code for a method `div` that throws an `ArithmetricException` if division by zero is attempted. It also prints "División por cero" to the console if an exception occurs. The bottom pane shows the test results: All 6 tests passed (100.00%) in 0.034 seconds. The output pane on the right shows repeated log entries for setup and teardown methods.

```
public int div(int a, int b) {
    try {
        if (b == 0) {
            throw new ArithmetricException("División por cero");
        }
        return a / b;
    } catch (ArithmetricException e) {
        System.out.println("División por cero");
    }
    return 0;
} 
```

Output Test Results x  
com.mycompany:mavenproject1:jar:1.0-SNAPSHOT (Unit) x  
Tests passed: 100.00 %  
All 6 tests passed: (0.034 s)

@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba  
División por cero  
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba

10.

The screenshot shows the NetBeans IDE interface. On the left, there is a code editor window displaying Java test code. The code defines a test method `addWhenNegativeThrowsException` that adds 6 and -1, expecting an `ArithmaticException`. The output window below shows the test results:

```
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Running MyCalculatorTest
@BeforeEach => setUpMyCalculator(): Se ha ejecutado una Calculadora ANTES DE CADA prueba
@AfterEach => tearDownMyCalculator(): Se ha eliminado la Calculadora DESPUÉS DE CADA prueba
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.027 s -- in MyCalculatorTest

Results:

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

-----
BUILD SUCCESS
-----
Total time: 0.981 s
Finished at: 2025-05-12T10:27:14+02:00
-----
```

11.

(No se como testear con coverage en NetBeans)