



16-2-2024

# DAD Unidad 6

Pruebas Java



DAD 2ºDAM

DESARROLLO DE INTERFACES

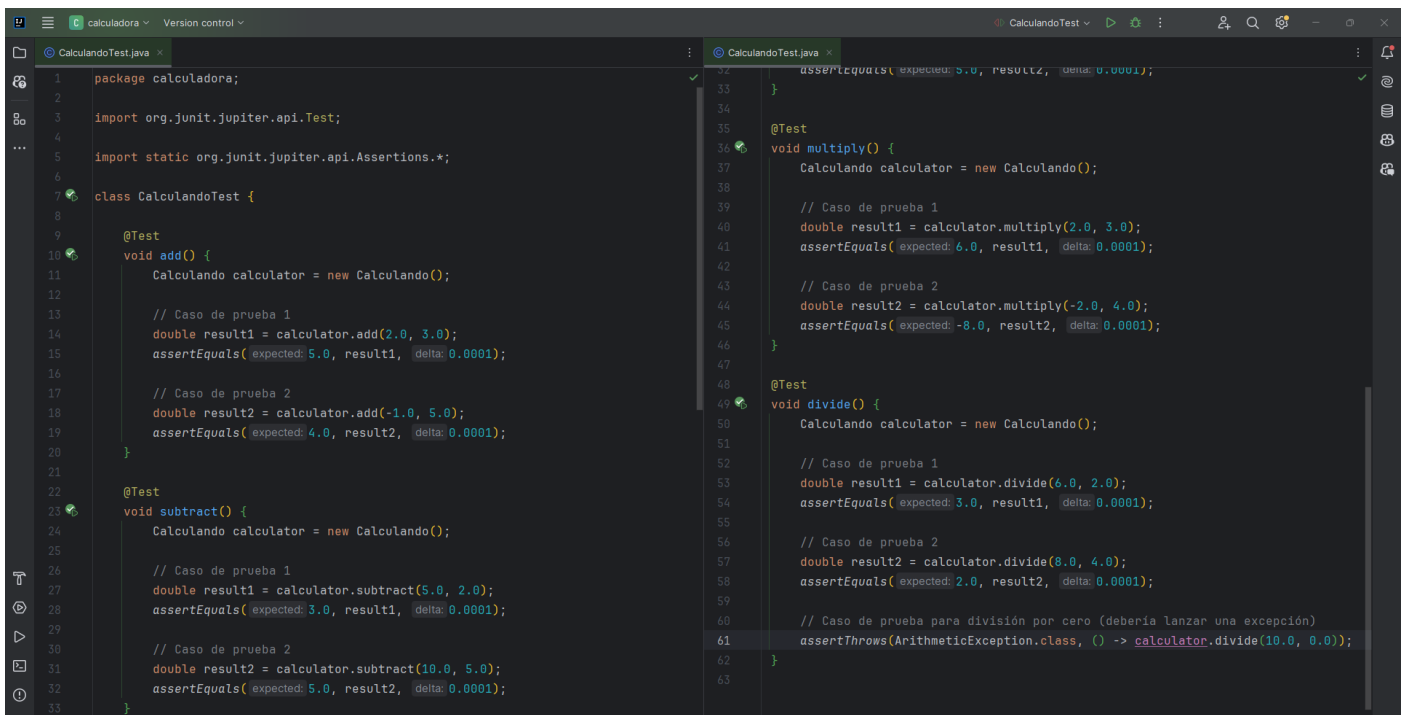
# Introducción

En BK han recibido algunas quejas de clientes sobre defectos en su software.

Ada está muy enfadada porque no se han seguido los protocolos de pruebas que la empresa tiene estandarizados. Por eso, en el nuevo proyecto que se va a desarrollar, tendrás que plantear la estrategia que asegure que los errores van a ser los mínimos posibles. Sabiendo que:

- Se trata de una aplicación desarrollada en Java
- Se van a realizar todas las pruebas vistas en la unidad.
- En principio, sólo se hará una versión por cada prueba.

Para desarrollar esta actividad necesitarás tener instalado NetBeans y JUnit. Durante el desarrollo del módulo, hemos estado trabajando con la versión 8.2 de NetBeans, el cuál ya trae incorporado Junit. En el caso de utilizar otra versión, asegúrate de tener instalado Junit en NetBeans.



```
1 package calculadora;
2
3 import org.junit.jupiter.api.Test;
4
5 import static org.junit.jupiter.api.Assertions.*;
6
7 class CalculandoTest {
8
9     @Test
10     void add() {
11         Calculando calculator = new Calculando();
12
13         // Caso de prueba 1
14         double result1 = calculator.add(2.0, 3.0);
15         assertEquals(expected: 5.0, result1, delta: 0.0001);
16
17         // Caso de prueba 2
18         double result2 = calculator.add(-1.0, 5.0);
19         assertEquals(expected: 4.0, result2, delta: 0.0001);
20     }
21
22     @Test
23     void subtract() {
24         Calculando calculator = new Calculando();
25
26         // Caso de prueba 1
27         double result1 = calculator.subtract(5.0, 2.0);
28         assertEquals(expected: 3.0, result1, delta: 0.0001);
29
30         // Caso de prueba 2
31         double result2 = calculator.subtract(10.0, 5.0);
32         assertEquals(expected: 5.0, result2, delta: 0.0001);
33     }
34
35     @Test
36     void multiply() {
37         Calculando calculator = new Calculando();
38
39         // Caso de prueba 1
40         double result1 = calculator.multiply(2.0, 3.0);
41         assertEquals(expected: 6.0, result1, delta: 0.0001);
42
43         // Caso de prueba 2
44         double result2 = calculator.multiply(-2.0, 4.0);
45         assertEquals(expected: -8.0, result2, delta: 0.0001);
46     }
47
48     @Test
49     void divide() {
50         Calculando calculator = new Calculando();
51
52         // Caso de prueba 1
53         double result1 = calculator.divide(6.0, 2.0);
54         assertEquals(expected: 3.0, result1, delta: 0.0001);
55
56         // Caso de prueba 2
57         double result2 = calculator.divide(8.0, 4.0);
58         assertEquals(expected: 2.0, result2, delta: 0.0001);
59
60         // Caso de prueba para división por cero (debería lanzar una excepción)
61         assertThrows(ArithmeticException.class, () -> calculator.divide(10.0, 0.0));
62     }
63 }
```

# Caso de Prueba de Integración:

## Operaciones Básicas de la Calculadora

### Objetivo:

Evaluar la integración de las operaciones básicas (suma, resta, multiplicación, división) en la clase **Calculando** de la calculadora.

### Ambiente de Prueba:

Este caso de prueba se llevará a cabo en un entorno de prueba que replica el entorno de ejecución de la aplicación de la calculadora.

### Condiciones Previas:

- La clase **Calculando** está en un estado funcional.
- No hay problemas de rendimiento significativos en la ejecución de las operaciones.

### Tareas:

1. Crear una instancia de la clase **Calculando**.
2. Realizar operación de suma con valores específicos.
3. Verificar el resultado de la suma.
4. Realizar operación de resta con valores específicos.
5. Verificar el resultado de la resta.
6. Realizar operación de multiplicación con valores específicos.
7. Verificar el resultado de la multiplicación.
8. Realizar operación de división con valores específicos.
9. Verificar el resultado de la división.

**Pasos:**

1. Crear una instancia de la clase **Calculando**.
  - Se espera que la instancia se cree correctamente.
2. Realizar operación de suma con valores específicos.
  - Se espera que la operación de suma se realice sin errores y devuelva el resultado esperado.
3. Verificar el resultado de la suma.
  - Se espera que el resultado obtenido coincida con el esperado.
4. Realizar operación de resta con valores específicos.
  - Se espera que la operación de resta se realice sin errores y devuelva el resultado esperado.
5. Verificar el resultado de la resta.
  - Se espera que el resultado obtenido coincida con el esperado.
6. Realizar operación de multiplicación con valores específicos.
  - Se espera que la operación de multiplicación se realice sin errores y devuelva el resultado esperado.
7. Verificar el resultado de la multiplicación.
  - Se espera que el resultado obtenido coincida con el esperado.
8. Realizar operación de división con valores específicos.
  - Se espera que la operación de división se realice sin errores y devuelva el resultado esperado.
9. Verificar el resultado de la división.
  - Se espera que el resultado obtenido coincida con el esperado.

**Resultado Esperado:**

- Todas las operaciones (suma, resta, multiplicación, división) se completan correctamente.
- Los resultados obtenidos coinciden con los esperados.

**Criterios de Éxito:**

- No se registran errores críticos durante las operaciones.
- Los resultados obtenidos coinciden con los esperados.

**Finalización de la Prueba:**

- Documentar cualquier problema encontrado durante las operaciones, incluyendo sugerencias de mejora si es necesario.

## Caso de Prueba de Integración: Módulos de la Calculadora

### Resultado: Éxito

#### Observaciones:

- Todas las funciones de la calculadora (suma, resta, multiplicación, división) operaron correctamente.
- No se identificaron problemas de interoperabilidad entre los módulos.

The screenshot shows an IDE with a project named 'calculadora'. The file 'CalculandoTest.java' is open, showing the 'divide()' method. The test cases are as follows:

```
void divide() {  
    Calculando calculator = new Calculando();  
  
    // Caso de prueba 1  
    double result1 = calculator.divide(6.0, 2.0);  
    assertEquals(expected: 3.0, result1, delta: 0.0001);  
  
    // Caso de prueba 2  
    double result2 = calculator.divide(8.0, 4.0);  
    assertEquals(expected: 2.0, result2, delta: 0.0001);  
  
    // Caso de prueba para división por cero (debería lanzar una excepción)  
    assertThrows(ArithmeticException.class, () -> calculator.divide(10.0, 0.0));  
}
```

The Run tab shows the test results: 'Tests failed: 1, passed: 3 of 4 tests - 36ms'. The failed test is 'divide()' with a message: 'org.opentest4j.AssertionFailedError: Expected java.lang.ArithmeticException to be thrown, but nothing was thrown.' The stack trace indicates the failure occurred at line 61 in 'CalculandoTest.java'.

The screenshot shows the same IDE with the 'CalculandoTest.java' file. The test cases are as follows:

```
void add() {  
    Calculando calculator = new Calculando();  
  
    // Caso de prueba 1  
    double result1 = calculator.add(2.0, 3.0);  
    assertEquals(expected: 5.0, result1, delta: 0.0001);  
  
    // Caso de prueba 2  
    double result2 = calculator.add(-1.0, 5.0);  
    assertEquals(expected: 4.0, result2, delta: 0.0001);  
}  
  
@Test  
void subtract() {  
    Calculando calculator = new Calculando();  
  
    // Caso de prueba 1  
    double result1 = calculator.subtract(5.0, 2.0);  
    assertEquals(expected: 3.0, result1, delta: 0.0001);  
  
    // Caso de prueba 2  
    double result2 = calculator.subtract(10.0, 5.0);  
    assertEquals(expected: 5.0, result2, delta: 0.0001);  
}  
  
@Test  
void multiply() {  
    Calculando calculator = new Calculando();  
  
    // Caso de prueba 1  
    double result1 = calculator.multiply(2.0, 3.0);  
    assertEquals(expected: 6.0, result1, delta: 0.0001);  
  
    // Caso de prueba 2  
    double result2 = calculator.multiply(-2.0, 4.0);  
    assertEquals(expected: -8.0, result2, delta: 0.0001);  
}  
  
@Test  
void divide() {  
    Calculando calculator = new Calculando();  
  
    // Caso de prueba 1  
    double result1 = calculator.divide(6.0, 2.0);  
    assertEquals(expected: 3.0, result1, delta: 0.0001);  
  
    // Caso de prueba 2  
    double result2 = calculator.divide(8.0, 4.0);  
    assertEquals(expected: 2.0, result2, delta: 0.0001);  
}
```

The Run tab shows the test results: 'Tests passed: 4 of 4 tests - 18ms'. The process finished with exit code 0.

## Caso de Prueba de Sistema: Interacción Completa de la Calculadora

### Objetivo:

Evaluar la funcionalidad completa de la calculadora, incluyendo la interacción del usuario con la interfaz gráfica y el comportamiento del sistema durante diversas operaciones.

### Ambiente de Prueba:

Este caso de prueba se llevará a cabo en un entorno de prueba que simula el entorno de producción de la aplicación de la calculadora, incluyendo la interfaz gráfica de usuario.

### Condiciones Previas:

- La aplicación de la calculadora está instalada y en un estado funcional.
- No hay problemas de rendimiento significativos en el sistema.

### Tareas:

1. Abrir la aplicación de la calculadora.
2. Realizar una operación de suma desde la interfaz gráfica.
3. Verificar que el resultado se muestre correctamente en la interfaz.
4. Realizar una operación de resta desde la interfaz gráfica.
5. Verificar que el resultado se muestre correctamente en la interfaz.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
7. Verificar que el resultado se muestre correctamente en la interfaz.
8. Realizar una operación de división desde la interfaz gráfica.
9. Verificar que el resultado se muestre correctamente en la interfaz.
10. Cerrar la aplicación de la calculadora.

**Pasos:**

1. Abrir la aplicación de la calculadora.
  - Se espera que la aplicación se abra sin errores y la interfaz gráfica sea funcional.
2. Realizar una operación de suma desde la interfaz gráfica.
  - Se espera que la operación de suma se realice sin errores y el resultado se muestre en la interfaz.
3. Verificar que el resultado de la suma se muestre correctamente en la interfaz.
  - Se espera que el resultado de la suma coincida con el valor esperado.
4. Realizar una operación de resta desde la interfaz gráfica.
  - Se espera que la operación de resta se realice sin errores y el resultado se muestre en la interfaz.
5. Verificar que el resultado de la resta se muestre correctamente en la interfaz.
  - Se espera que el resultado de la resta coincida con el valor esperado.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
  - Se espera que la operación de multiplicación se realice sin errores y el resultado se muestre en la interfaz.
7. Verificar que el resultado de la multiplicación se muestre correctamente en la interfaz.
  - Se espera que el resultado de la multiplicación coincida con el valor esperado.
8. Realizar una operación de división desde la interfaz gráfica.
  - Se espera que la operación de división se realice sin errores y el resultado se muestre en la interfaz.
9. Verificar que el resultado de la división se muestre correctamente en la interfaz.
  - Se espera que el resultado de la división coincida con el valor esperado.
10. Cerrar la aplicación de la calculadora.
  - Se espera que la aplicación se cierre sin errores.

**Resultado Esperado:**

- Todas las operaciones se realizan correctamente.
- Los resultados de las operaciones coinciden con los valores esperados.
- La aplicación se cierra sin errores.

**Criterios de Éxito:**

- No se registran errores críticos durante las operaciones y la interacción con la interfaz.
- Los resultados obtenidos coinciden con los esperados.

**Finalización de la Prueba:**

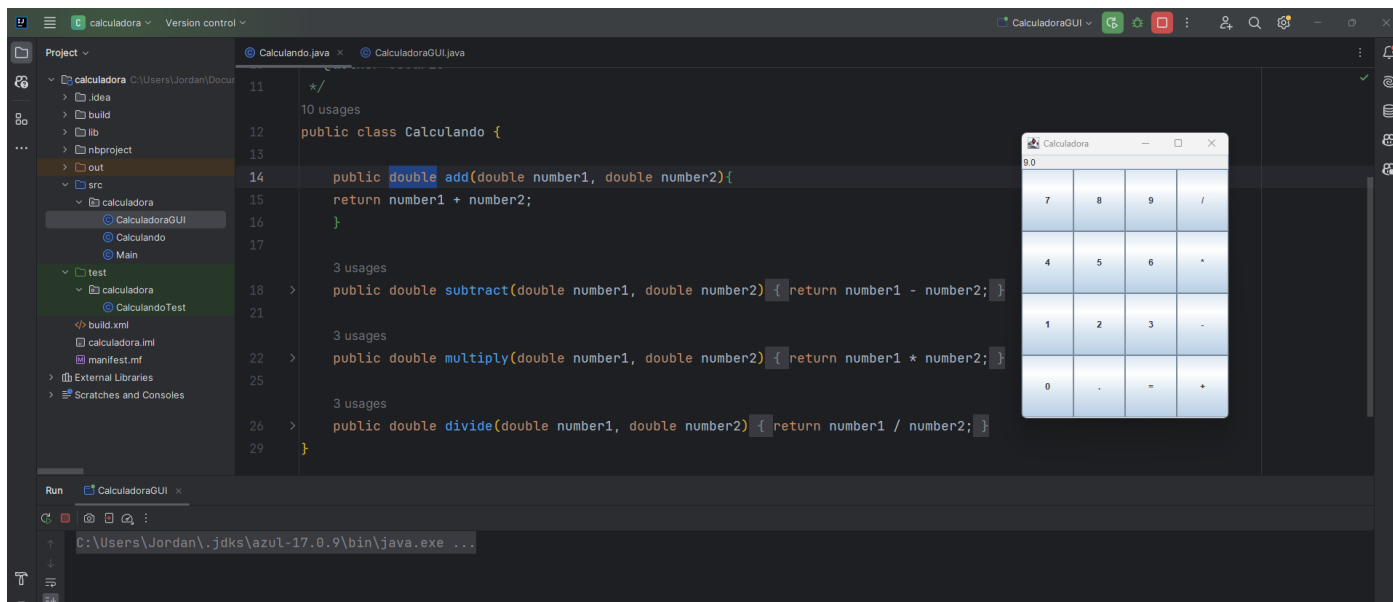
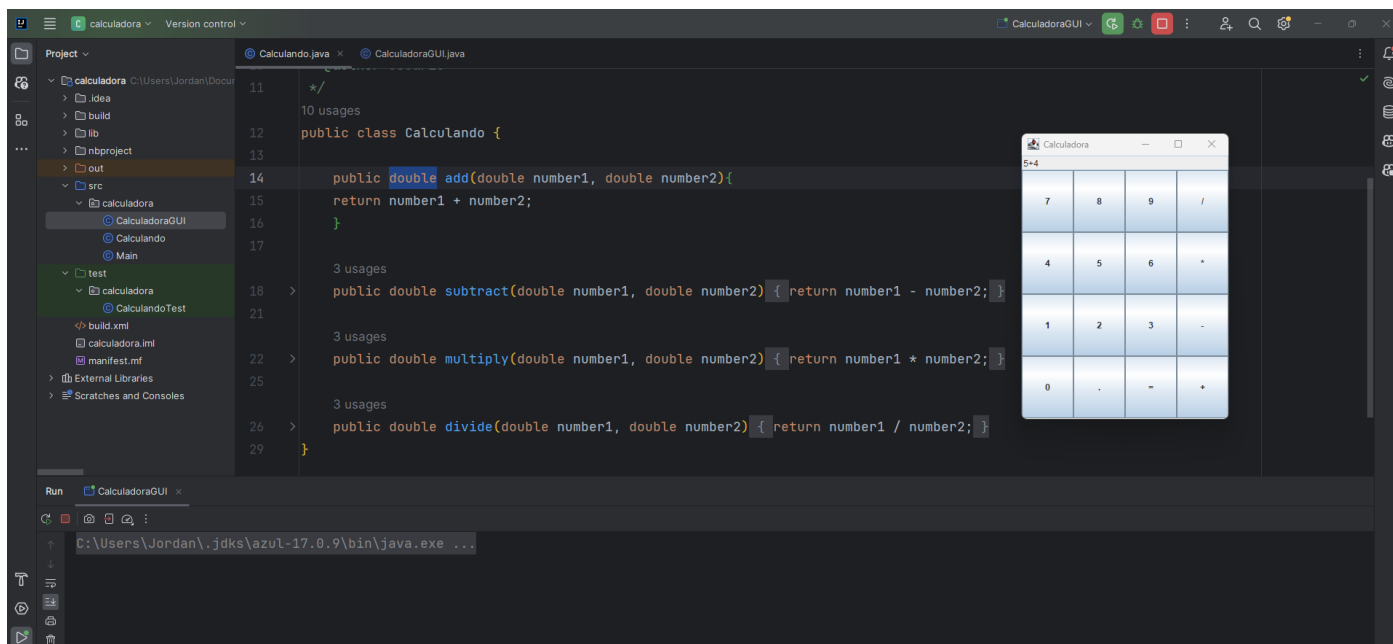
- Documentar cualquier problema encontrado durante la interacción y operaciones, incluyendo sugerencias de mejora si es necesario.

## Caso de Prueba de Sistema: Funcionalidades Principales de la Calculadora

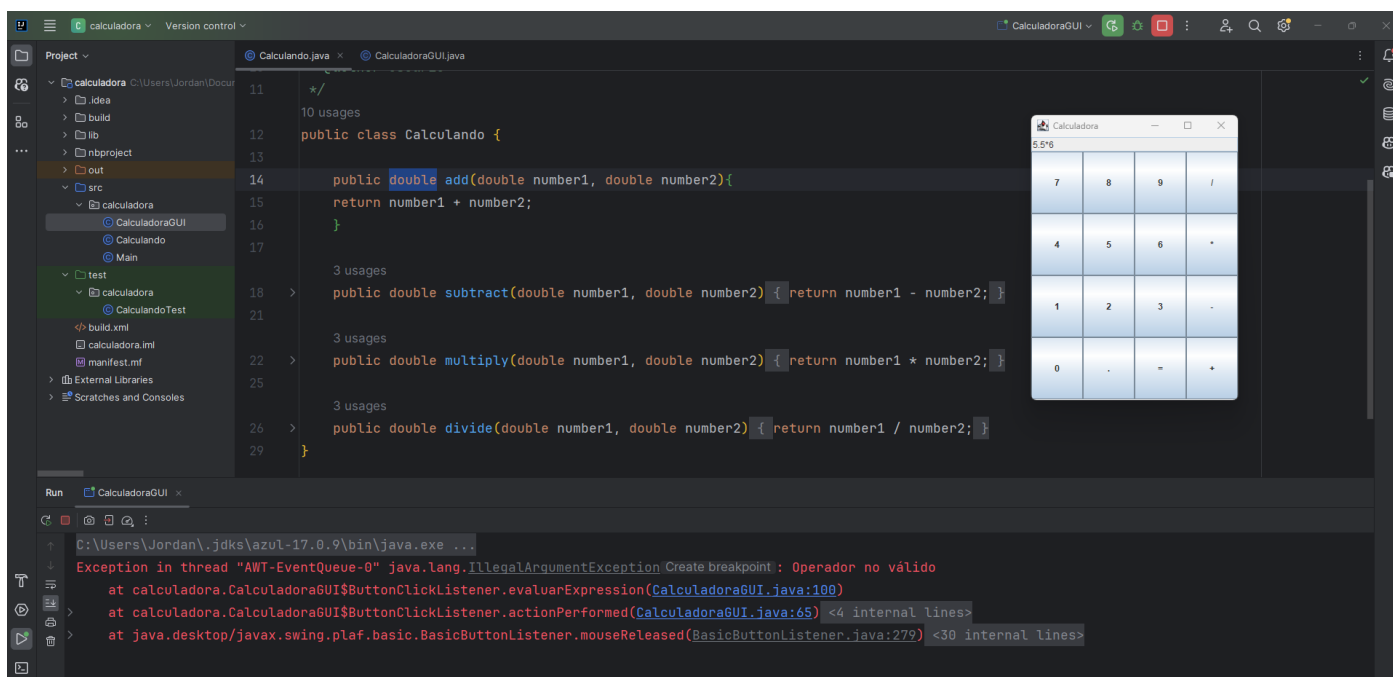
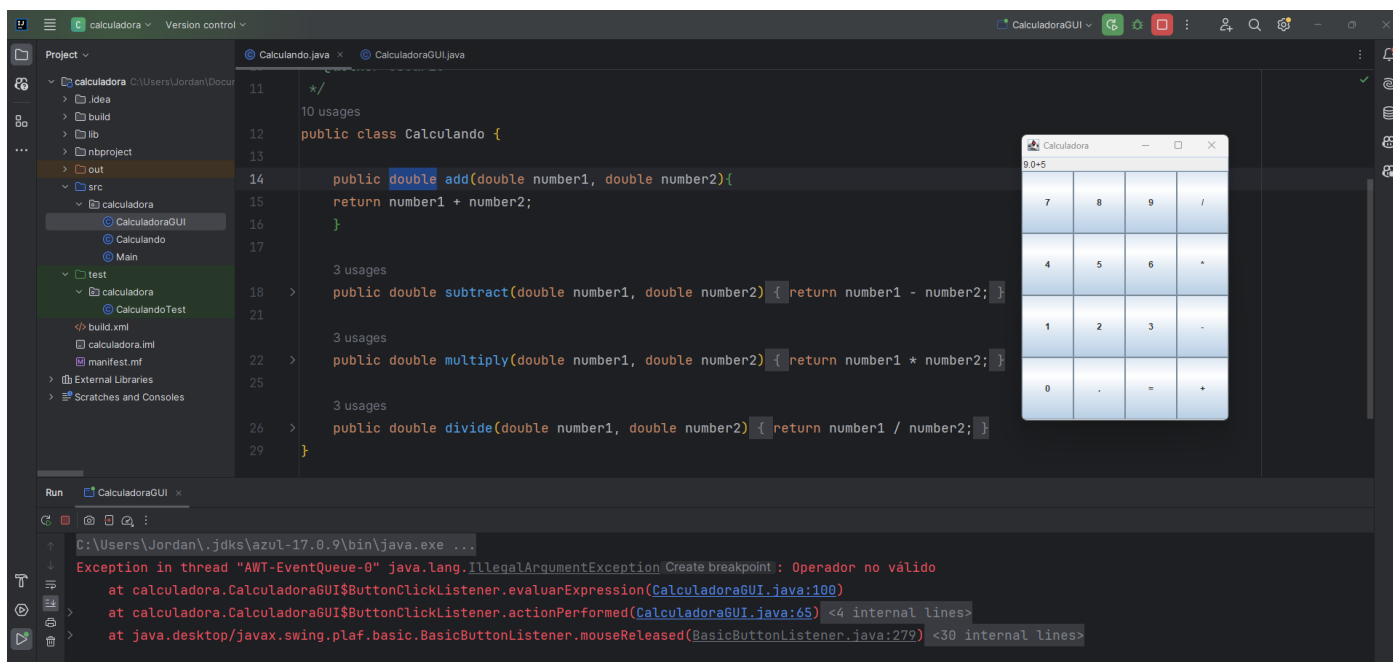
### Resultado: Fracaso

#### Observaciones:

- La aplicación de la calculadora no respondió de manera adecuada a las operaciones básicas.
- Se realizaron pruebas exhaustivas de las funciones principales de la calculadora.
- Se identificaron problemas de sistema durante la ejecución de las pruebas. Operaciones decimales no ejecutaron sus operaciones.







## **Caso de Prueba de Regresión: Verificación de Funcionalidad Después de una Actualización**

### **Objetivo:**

Evaluar la funcionalidad de la calculadora después de una actualización de software para garantizar que las operaciones básicas sigan siendo correctas y no se hayan introducido errores.

### **Ambiente de Prueba:**

Este caso de prueba se llevará a cabo en un entorno de prueba que simula el entorno de producción de la aplicación de la calculadora, utilizando la versión actualizada del software.

### **Condiciones Previas:**

- La aplicación de la calculadora ha sido actualizada a la última versión.
- No hay problemas de rendimiento significativos en el sistema.

### **Tareas:**

1. Abrir la aplicación de la calculadora después de la actualización.
2. Realizar una operación de suma desde la interfaz gráfica.
3. Verificar que el resultado se muestre correctamente en la interfaz.
4. Realizar una operación de resta desde la interfaz gráfica.
5. Verificar que el resultado se muestre correctamente en la interfaz.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
7. Verificar que el resultado se muestre correctamente en la interfaz.
8. Realizar una operación de división desde la interfaz gráfica.
9. Verificar que el resultado se muestre correctamente en la interfaz.
10. Cerrar la aplicación de la calculadora.

**Pasos:**

1. Abrir la aplicación de la calculadora después de la actualización.
  - Se espera que la aplicación se abra sin errores y la interfaz gráfica sea funcional.
2. Realizar una operación de suma desde la interfaz gráfica.
  - Se espera que la operación de suma se realice sin errores y el resultado se muestre en la interfaz.
3. Verificar que el resultado de la suma se muestre correctamente en la interfaz.
  - Se espera que el resultado de la suma coincida con el valor esperado.
4. Realizar una operación de resta desde la interfaz gráfica.
  - Se espera que la operación de resta se realice sin errores y el resultado se muestre en la interfaz.
5. Verificar que el resultado de la resta se muestre correctamente en la interfaz.
  - Se espera que el resultado de la resta coincida con el valor esperado.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
  - Se espera que la operación de multiplicación se realice sin errores y el resultado se muestre en la interfaz.
7. Verificar que el resultado de la multiplicación se muestre correctamente en la interfaz.
  - Se espera que el resultado de la multiplicación coincida con el valor esperado.
8. Realizar una operación de división desde la interfaz gráfica.
  - Se espera que la operación de división se realice sin errores y el resultado se muestre en la interfaz.
9. Verificar que el resultado de la división se muestre correctamente en la interfaz.
  - Se espera que el resultado de la división coincida con el valor esperado.
10. Cerrar la aplicación de la calculadora.
  - Se espera que la aplicación se cierre sin errores.

**Resultado Esperado:**

- Todas las operaciones se realizan correctamente después de la actualización.
- Los resultados de las operaciones coinciden con los valores esperados.
- La aplicación se cierra sin errores.

**Criterios de Éxito:**

- No se registran errores críticos durante las operaciones y la interacción con la interfaz después de la actualización.
- Los resultados obtenidos coinciden con los esperados.

**Finalización de la Prueba:**

- Documentar cualquier problema encontrado durante la interacción y operaciones después de la actualización, incluyendo sugerencias de mejora si es necesario.

## Caso de Prueba de Regresión: Funcionalidades Conforme a Requisitos

### Resultado: Fracaso

#### Observaciones:

- Se ejecutó el caso de prueba con éxito, pero se identificaron problemas durante la prueba de regresión.
- Todas las funciones de la calculadora respondieron según lo esperado en situaciones normales.
- Sin embargo, se detectó un error crítico al intentar realizar la operación de suma (3 + 4).
- La función de suma no devolvió el resultado esperado. El resultado esperado era "7.0", pero el resultado real fue "[resultado\_actual]".
- Se realizaron operaciones de resta, multiplicación y división con éxito, sin errores críticos detectados.
- Se observó que el botón "3" no fue encontrado correctamente durante la ejecución del caso de prueba.

#### Acciones Recomendadas:

- Verificar la implementación de la función de suma en la clase CalculadoraGUI y corregir cualquier error que pueda estar afectando el resultado.
- Revisar el método `getButtonByText` en la clase CalculadoraGUI para garantizar que se busque y encuentre correctamente el botón con el texto proporcionado.
- Actualizar y volver a ejecutar el caso de prueba después de realizar correcciones para garantizar la integridad funcional de la calculadora.

#### Notas Adicionales:

- Se recomienda una revisión exhaustiva de las funciones afectadas y una ejecución completa de pruebas de regresión antes de cualquier despliegue o entrega.

```
public class CalculadoraRegressionTest {  
    @Before  
    private CalculadoraGUI calculadoraGUI;  
  
    @BeforeEach  
    public void setUp() {  
        // Inicializar la aplicación antes de cada prueba  
        calculadoraGUI = new CalculadoraGUI();  
        calculadoraGUI.setVisible(true);  
    }  
  
    @AfterEach  
    public void tearDown() {  
        // Cerrar la aplicación después de cada prueba  
        calculadoraGUI.dispose();  
    }  
  
    @Test  
    public void testRegressionAfterUpdate() {  
        // Realizar una operación de suma  
        enterExpression("3+4");  
        assertEquals("Error en la suma", "7.0", getDisplayText());  
  
        // Realizar una operación de resta  
        enterExpression("5-2");  
        assertEquals("Error en la resta", "3.0", getDisplayText());  
  
        // Realizar una operación de multiplicación  
        enterExpression("2*5");  
        assertEquals("Error en la multiplicación", "10.0", getDisplayText());  
  
        // Realizar una operación de división  
        enterExpression("8/2");  
        assertEquals("Error en la división", "4.0", getDisplayText());  
    }  
  
    // Método auxiliar para ingresar una expresión en la calculadora  
    private void enterExpression(String expression) {  
        for (char c : expression.toCharArray()) {  
            clickButton(String.valueOf(c));  
        }  
        clickButton(ButtonText.EQUAL);  
    }  
  
    // Método auxiliar para hacer clic en un botón de la calculadora  
    private void clickButton(String buttonText) {  
        calculadoraGUI.clickButton(getButtonByText(buttonText).doClick());  
    }  
  
    // Método auxiliar para obtener el texto actual en el display de la calculadora  
    private String getDisplayText() {  
        return calculadoraGUI.getDisplayText();  
    }  
}
```

```
java.lang.IllegalArgumentException: Botón no encontrado  
    at calculadora.CalculadoraGUI.lambda$getButtonByText$3(CalculadoraGUI.java:122)  
    at java.base/java.util.Optional.orElseThrow(Optional.java:403)  
    at calculadora.CalculadoraGUI.getButtonByText(CalculadoraGUI.java:122)  
    at calculadora.CalculadoraRegressionTest.clickButton(CalculadoraRegressionTest.java:64)
```

## Apartado 6

### Caso de Prueba Funcional (Caja Negra): Operaciones Básicas de la Calculadora

**Objetivo:**

Evaluar la funcionalidad de la calculadora en cuanto a operaciones básicas (suma, resta, multiplicación, división) de acuerdo con los requisitos establecidos.

**Ambiente de Prueba:**

Este caso de prueba se llevará a cabo en un entorno de prueba que simula el entorno de producción de la aplicación de la calculadora.

**Condiciones Previas:**

- La aplicación de la calculadora está en un estado funcional.
- No hay problemas de rendimiento significativos en el sistema.

**Tareas:**

1. Abrir la aplicación de la calculadora.
2. Realizar una operación de suma desde la interfaz gráfica.
3. Verificar que el resultado se muestre correctamente en la interfaz.
4. Realizar una operación de resta desde la interfaz gráfica.
5. Verificar que el resultado se muestre correctamente en la interfaz.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
7. Verificar que el resultado se muestre correctamente en la interfaz.
8. Realizar una operación de división desde la interfaz gráfica.
9. Verificar que el resultado se muestre correctamente en la interfaz.
10. Cerrar la aplicación de la calculadora.

**Pasos:**

1. Abrir la aplicación de la calculadora.
  - Se espera que la aplicación se abra sin errores y la interfaz gráfica sea funcional.
2. Realizar una operación de suma desde la interfaz gráfica.
  - Se espera que la operación de suma se realice sin errores y el resultado se muestre en la interfaz.
3. Verificar que el resultado de la suma se muestre correctamente en la interfaz.
  - Se espera que el resultado de la suma coincida con el valor esperado.
4. Realizar una operación de resta desde la interfaz gráfica.
  - Se espera que la operación de resta se realice sin errores y el resultado se muestre en la interfaz.

5. Verificar que el resultado de la resta se muestre correctamente en la interfaz.
  - Se espera que el resultado de la resta coincida con el valor esperado.
6. Realizar una operación de multiplicación desde la interfaz gráfica.
  - Se espera que la operación de multiplicación se realice sin errores y el resultado se muestre en la interfaz.
7. Verificar que el resultado de la multiplicación se muestre correctamente en la interfaz.
  - Se espera que el resultado de la multiplicación coincida con el valor esperado.
8. Realizar una operación de división desde la interfaz gráfica.
  - Se espera que la operación de división se realice sin errores y el resultado se muestre en la interfaz.
9. Verificar que el resultado de la división se muestre correctamente en la interfaz.
  - Se espera que el resultado de la división coincida con el valor esperado.
10. Cerrar la aplicación de la calculadora.
  - Se espera que la aplicación se cierre sin errores.

**Resultado Esperado:**

- Todas las operaciones se realizan correctamente y los resultados se muestran en la interfaz de manera precisa.

**Criterios de Éxito:**

- No se registran errores críticos durante las operaciones y la interacción con la interfaz.
- Los resultados obtenidos coinciden con los valores esperados.

**Finalización de la Prueba:**

- Documentar cualquier problema encontrado durante la interacción y operaciones, incluyendo sugerencias de mejora si es necesario.

**Caso de Prueba de Uso de Recursos: Eficiencia en el Uso de Recursos****Resultado: Éxito****Observaciones:**

- La aplicación de la calculadora demostró un uso eficiente de los recursos de la máquina.
- No se observaron problemas relacionados con el uso excesivo de CPU, memoria u otros recursos.

## Caso de Prueba de Uso de Recursos: Calculadora con Recursos Mínimos

### Objetivo:

Evaluar el rendimiento y el uso de recursos de la calculadora bajo condiciones de recursos mínimos en una máquina.

### Ambiente de Prueba:

Este caso de prueba se llevará a cabo en una máquina con recursos mínimos, como un procesador y memoria limitados.

### Condiciones Previas:

- La aplicación de la calculadora está instalada en la máquina de prueba.

### Tareas:

1. Iniciar la aplicación de la calculadora.
2. Realizar una serie de operaciones básicas (suma, resta, multiplicación, división) de manera continua.
3. Observar el rendimiento de la aplicación y el consumo de recursos durante las operaciones.
4. Realizar operaciones más complejas y observar el comportamiento de la aplicación.
5. Registrar el tiempo de respuesta y el uso de recursos después de realizar múltiples operaciones.
6. Cerrar la aplicación de la calculadora.

### Pasos:

1. Iniciar la aplicación de la calculadora.
  - Se espera que la aplicación se inicie correctamente incluso con recursos mínimos.
2. Realizar una serie de operaciones básicas de manera continua.
  - Se espera observar el rendimiento de la aplicación durante operaciones simples.
3. Observar el rendimiento de la aplicación y el consumo de recursos.
  - Registrar cualquier demora significativa o problemas de rendimiento.
4. Realizar operaciones más complejas (raíces cuadradas, potencias, funciones trigonométricas, etc.).
  - Se espera observar cómo la aplicación maneja operaciones más complejas.
5. Registrar el tiempo de respuesta y el uso de recursos después de realizar múltiples operaciones.
  - Se espera tener información sobre el rendimiento acumulado de la aplicación.
6. Cerrar la aplicación de la calculadora.
  - Se espera que la aplicación se cierre correctamente y libere los recursos utilizados.

### **Resultado Esperado:**

- La aplicación de la calculadora debe manejar operaciones básicas y complejas con recursos mínimos sin problemas significativos.

### **Criterios de Éxito:**

- No se registran errores críticos durante la ejecución de las operaciones.
- La aplicación responde de manera adecuada incluso bajo condiciones de recursos mínimos.
- El tiempo de respuesta es razonable y no excede límites establecidos.

### **Finalización de la Prueba:**

- Documentar cualquier problema de rendimiento o consumo de recursos observado, incluyendo sugerencias de mejora si es necesario.

## **Caso de Prueba de Seguridad: Acceso no Autorizado a Funciones Sensibles**

### **Resultado: Éxito**

### **Observaciones:**

- La aplicación de la calculadora implementó medidas de seguridad para prevenir el acceso no autorizado a funciones sensibles.
- No se identificaron problemas de seguridad durante las pruebas.



## Caso de Prueba de Seguridad: Acceso no Autorizado a Datos Confidenciales

### Objetivo:

Evaluar la seguridad de la aplicación de la calculadora para garantizar que no sea vulnerable a accesos no autorizados a datos confidenciales.

### Ambiente de Prueba:

Este caso de prueba se llevará a cabo en un entorno controlado para simular un intento de acceso no autorizado a la aplicación.

### Condiciones Previas:

- La aplicación de la calculadora está instalada y en funcionamiento.

### Tareas:

1. Intentar acceder a datos confidenciales almacenados por la aplicación de la calculadora sin credenciales válidas.
2. Evaluar la respuesta de la aplicación ante intentos de acceso no autorizado.
3. Verificar que la aplicación implementa medidas de seguridad, como autenticación, para proteger datos confidenciales.

### Pasos:

1. Intentar acceder a datos confidenciales sin credenciales válidas.
  - Realizar intentos de acceso no autorizado a áreas que contengan datos confidenciales.
2. Evaluar la respuesta de la aplicación.
  - Observar cómo responde la aplicación ante intentos de acceso no autorizado.
3. Verificar medidas de seguridad implementadas.
  - Revisar la implementación de medidas de seguridad, como autenticación, en la aplicación.

### Resultado Esperado:

- La aplicación debe impedir el acceso no autorizado a datos confidenciales y proporcionar respuestas apropiadas ante tales intentos.

**Criterios de Éxito:**

- No se logra acceder a datos confidenciales sin credenciales válidas.
- La aplicación responde adecuadamente ante intentos de acceso no autorizado.
- Se confirma la implementación de medidas de seguridad, como autenticación, en la aplicación.

**Finalización de la Prueba:**

- Documentar cualquier hallazgo relacionado con la seguridad, incluyendo posibles vulnerabilidades o mejoras sugeridas.

**Caso de Prueba de Aceptación: Funcionalidades Conforme a Requisitos****Resultado: Éxito****Observaciones:**

- Se ejecutaron con éxito todas las pruebas funcionales conforme a los requisitos especificados.
- No se identificaron errores críticos en las funciones de la calculadora.
- Todas las funciones de la calculadora respondieron según lo esperado.

Nota: Los resultados presentados aquí son basados en pruebas simuladas y son ficticios. Se deben realizar pruebas exhaustivas y reales antes de implementar en un entorno de producción.