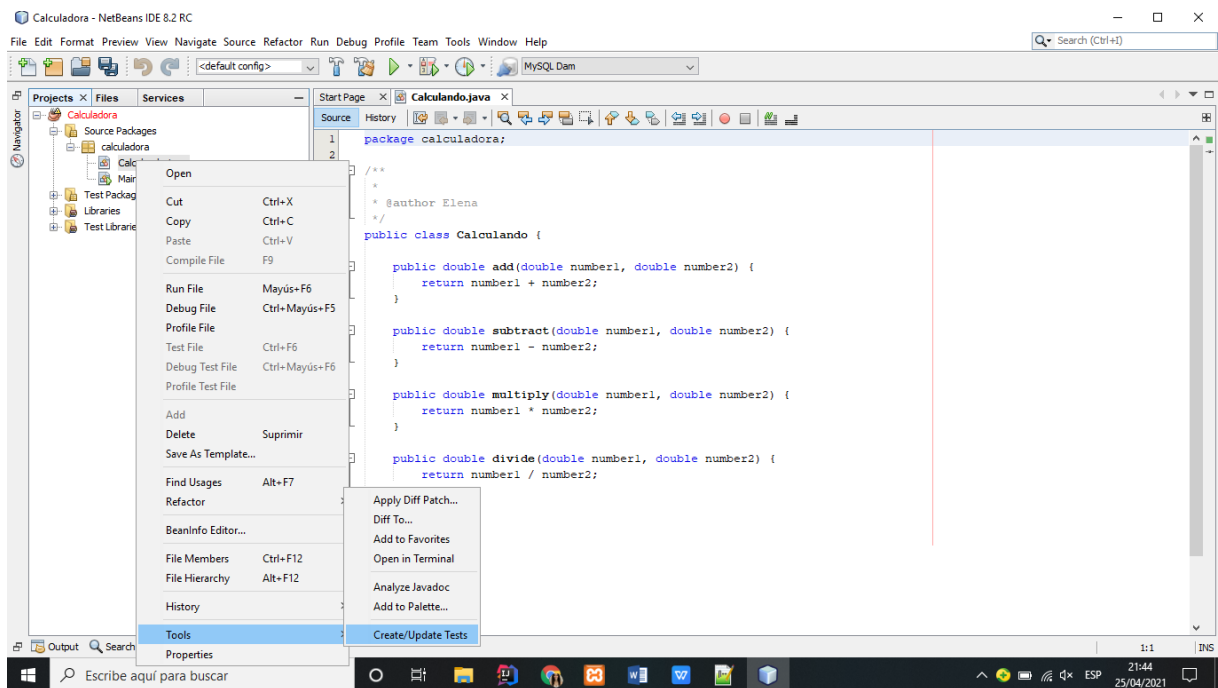


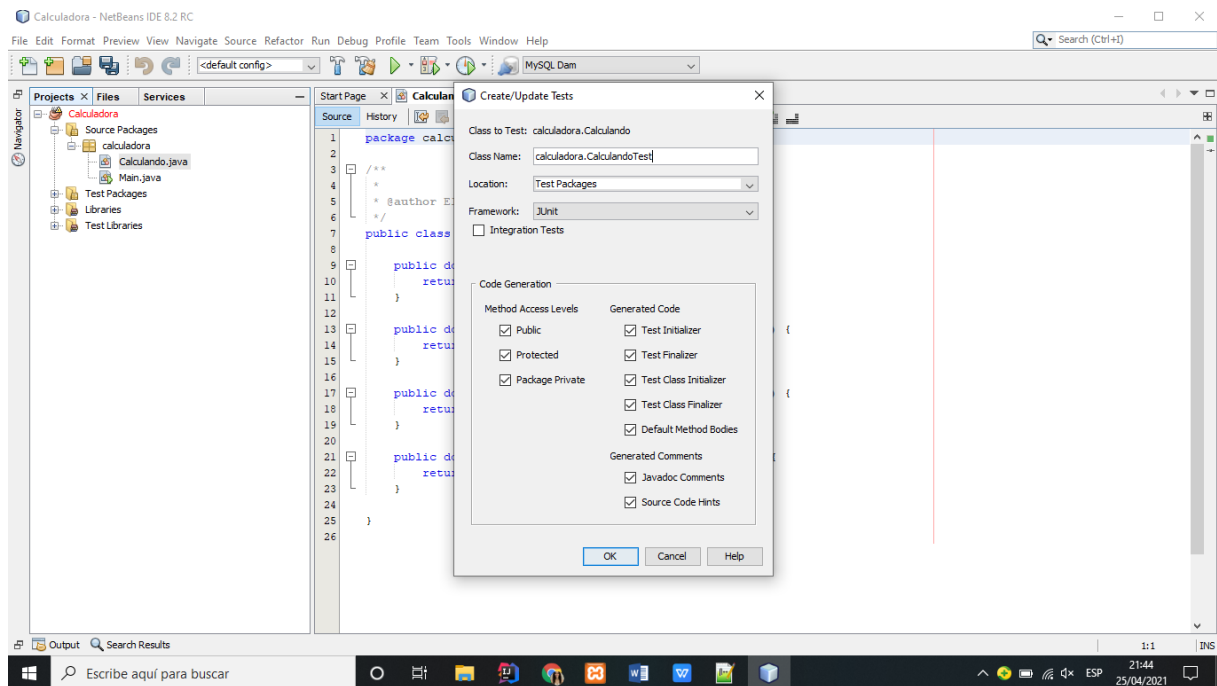
TAREA 8 – DESARROLLO DE INTERFACES

****NOTA: No me di cuenta de cambiar mi nombre en el proyecto y lo modifíco en mitad de la tarea en el punto 5**

Ejercicio 1

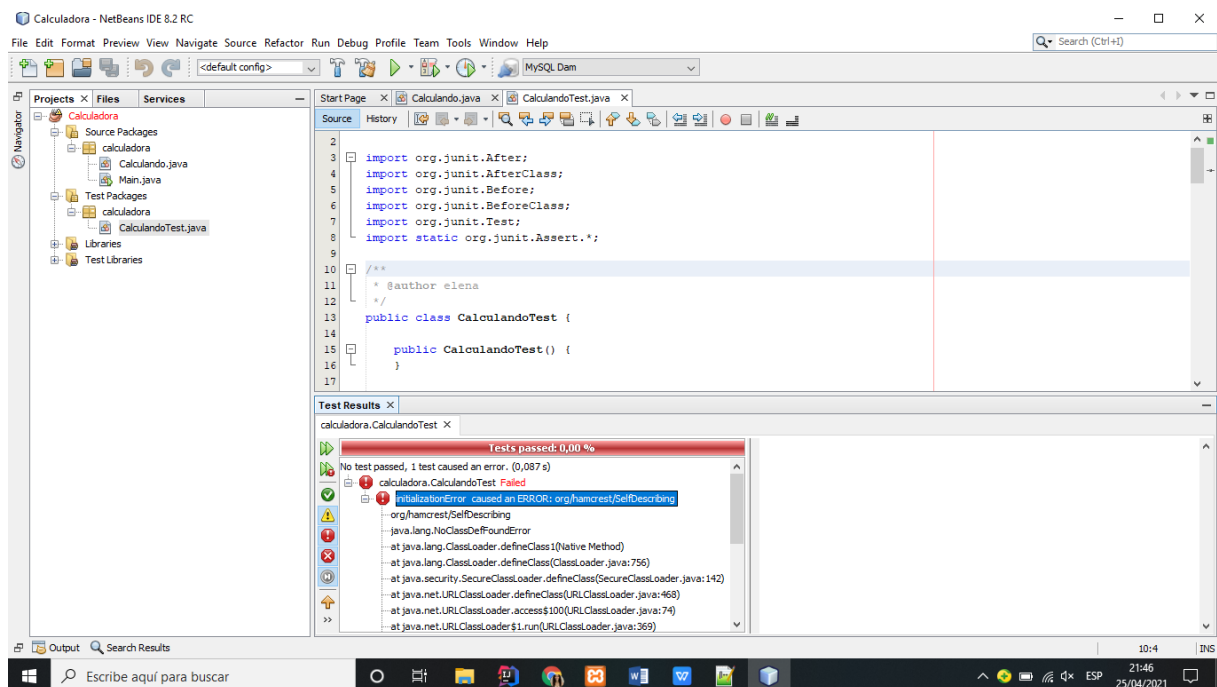
Descarga el proyecto Java y ábrelo con NetBeans. Observa los métodos definidos en la clase Calculando.java. Vamos a probar cada método de la clase con JUnit. Para ello, deberás de seleccionar la clase y en el menú Herramientas deberás de seleccionar la opción Create /update Tests. Nos aparecerá una ventana donde consta la clase a la que se le van a realizar las pruebas y la ubicación de las mismas. Seleccionaremos como Framework Junit y veremos que el código de la aplicación importa automáticamente el framework Junit. Como solución a este apartado deberás de aportar el código de la clase generado. (Calificación 1 punto)



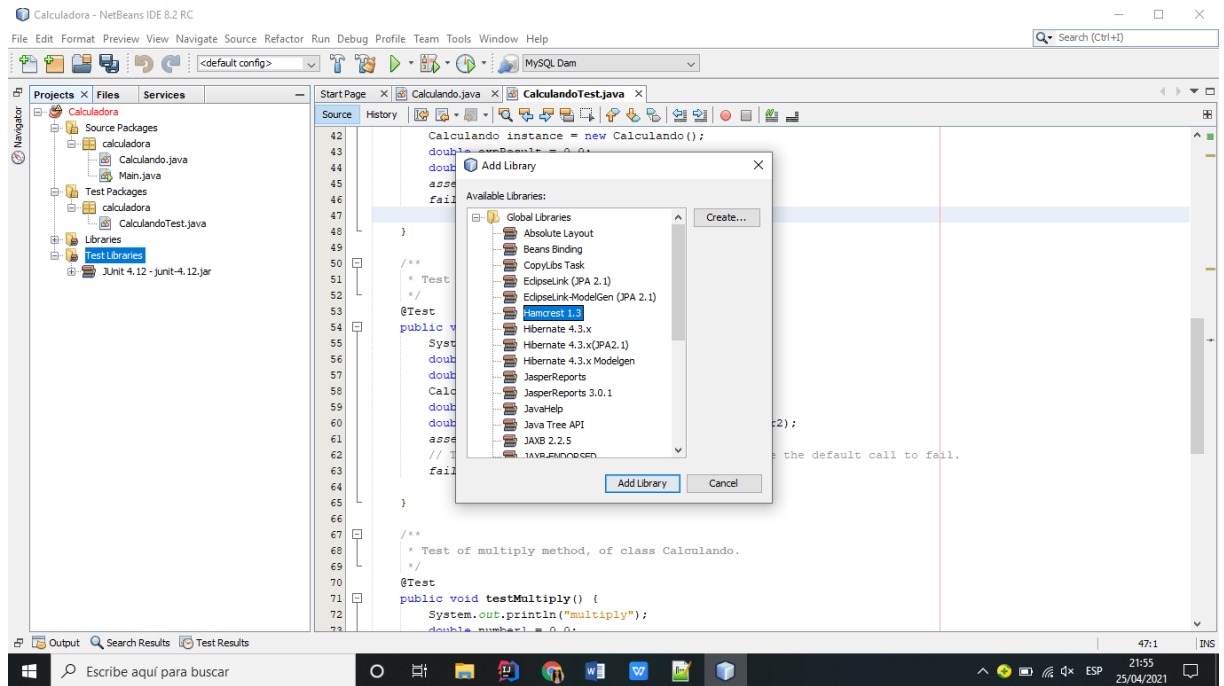


Ejercicio 2

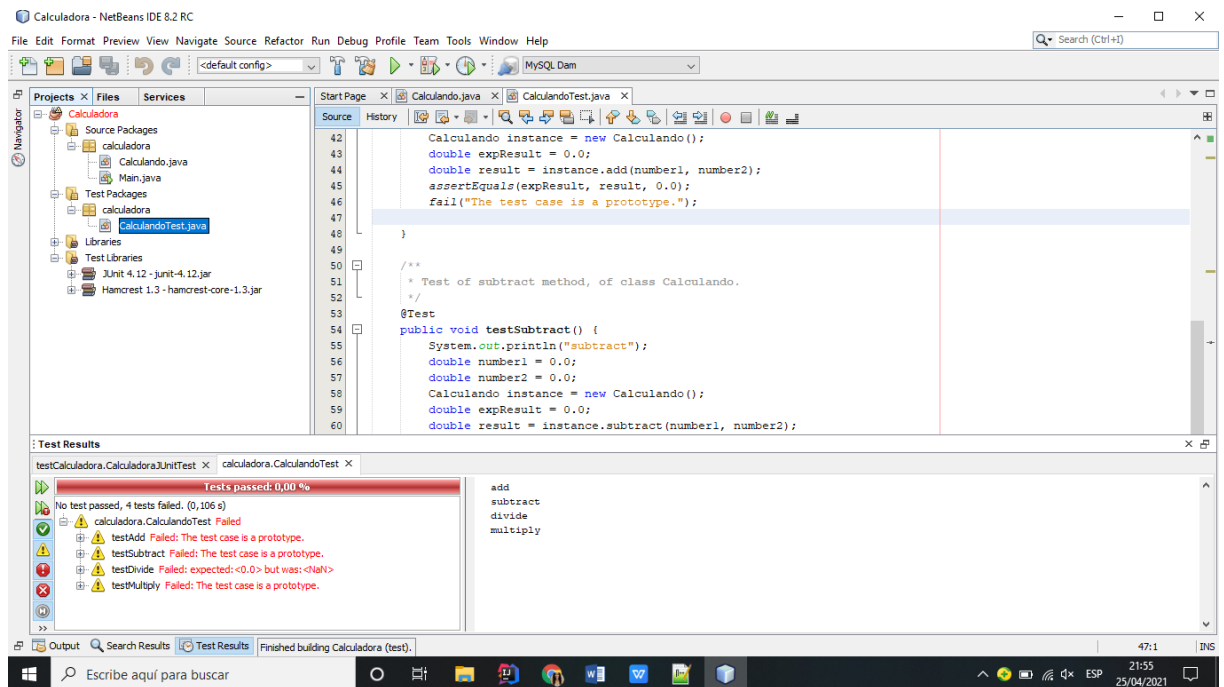
Selecciona la nueva clase de pruebas que has generado. Ejecútala. Realiza una captura de la ventana Test results como solución a este apartado. (Calificación 1 punto)



Nos falla porque le falta la librería Harmcrest. La cargamos y volvemos a ejecutar



Al añadir la dependencia vuelvo a ejecutar los test creados automáticamente y fallan



Ejercicio 3

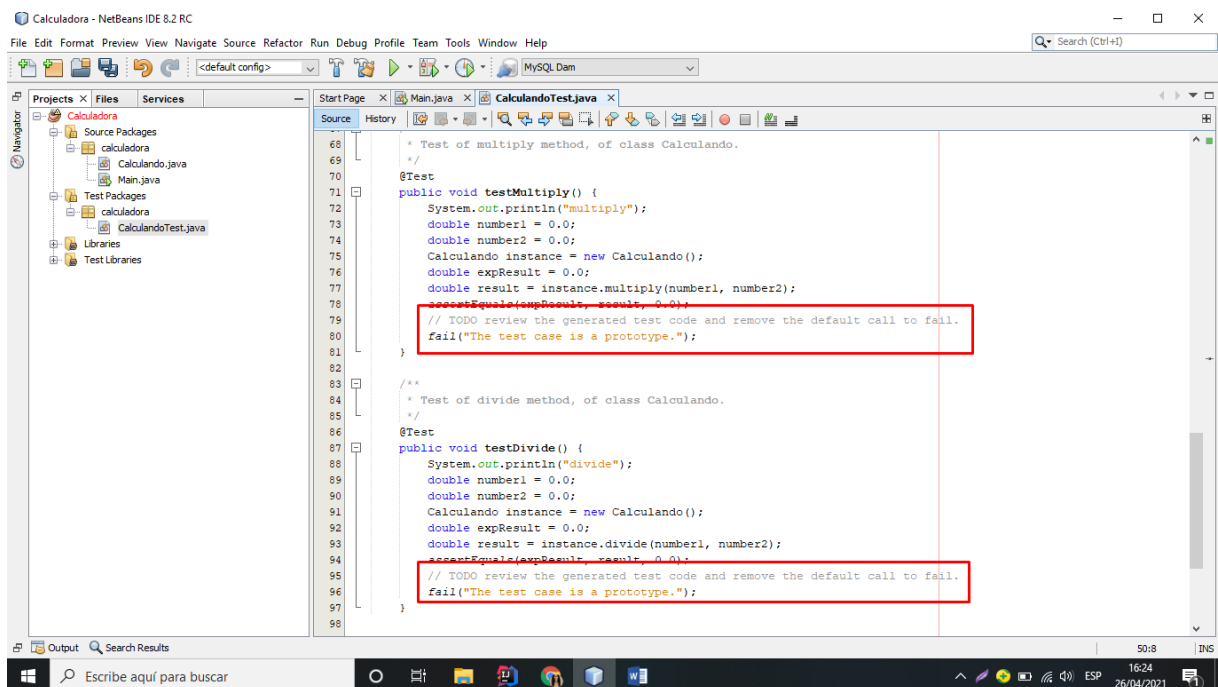
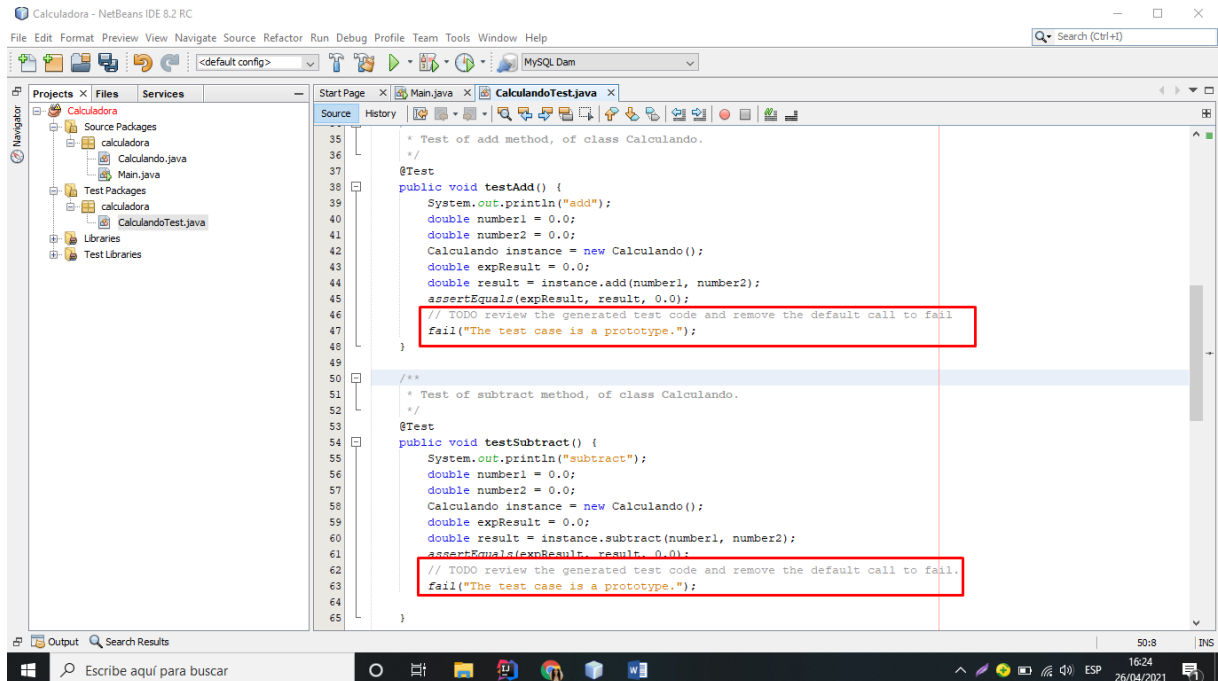
Accede al código de la clase de pruebas y elimina las líneas:

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");

que aparece al final de cada método. Como solución a este apartado deberás de entregar el código de la clase generado. (Calificación 1 punto).

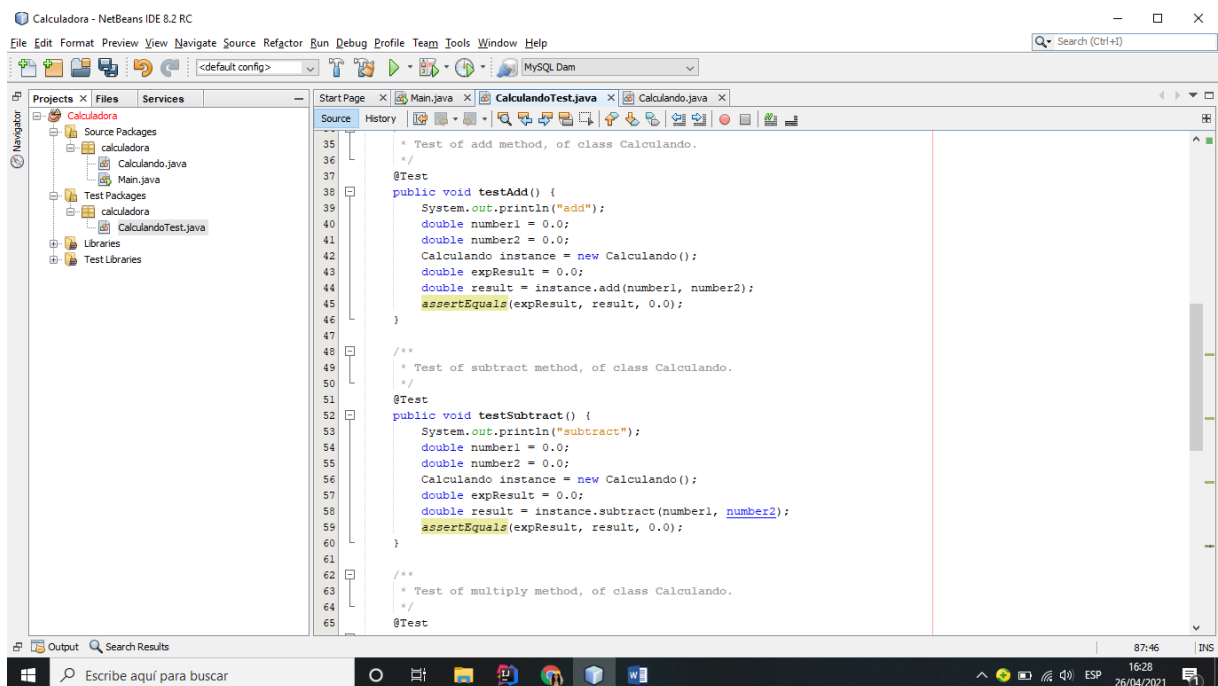
Netbeans autogenera el código de test de la clase Calculando que incluye los métodos add, subtract, multiply, divide.



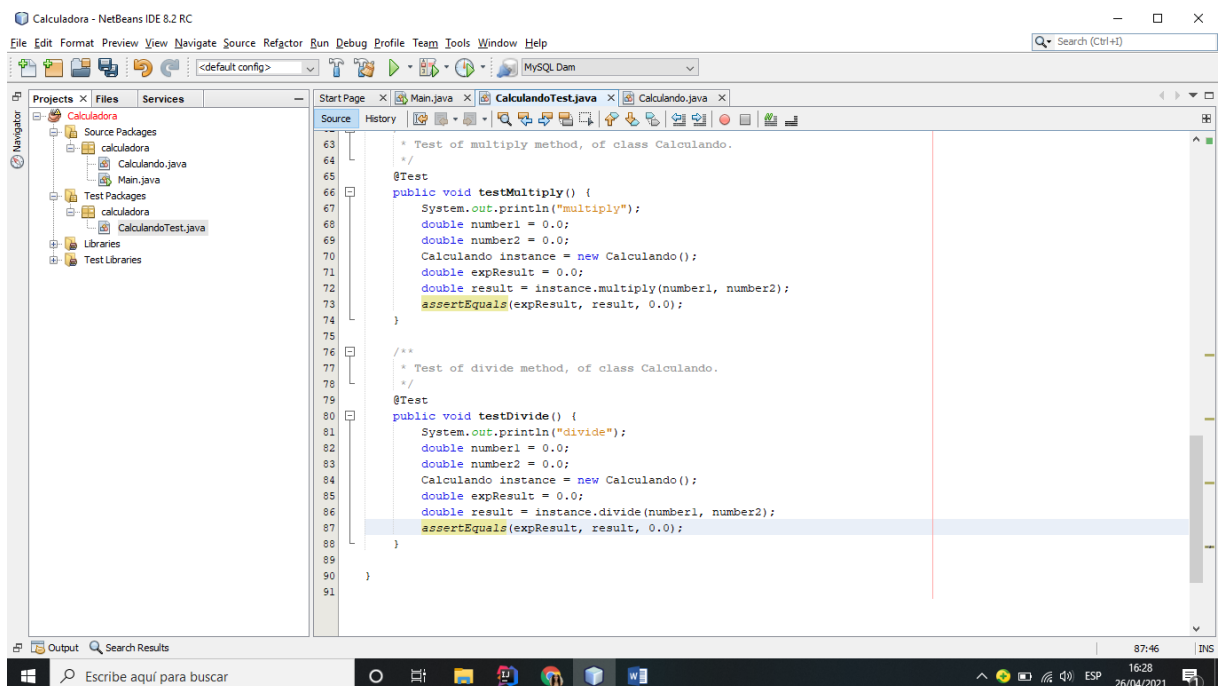
Eliminamos las líneas de los test autogenerados

// TODO review the generated test code and remove the default call to fail.

fail("The test case is a prototype.");



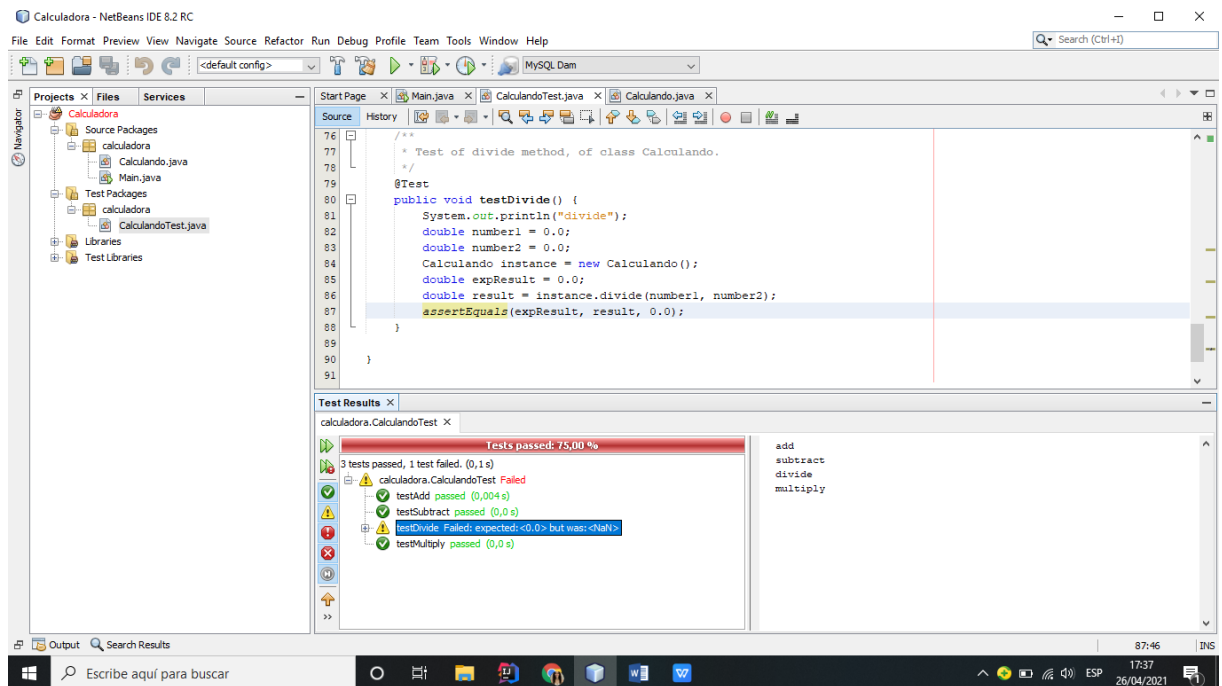
```
35  * Test of add method, of class Calculando.
36  */
37  @Test
38  public void testAdd() {
39      System.out.println("add");
40      double number1 = 0.0;
41      double number2 = 0.0;
42      Calculando instance = new Calculando();
43      double expectedResult = 0.0;
44      double result = instance.add(number1, number2);
45      assertEquals(expResult, result, 0.0);
46  }
47
48  /**
49   * Test of subtract method, of class Calculando.
50   */
51  @Test
52  public void testSubtract() {
53      System.out.println("subtract");
54      double number1 = 0.0;
55      double number2 = 0.0;
56      Calculando instance = new Calculando();
57      double expectedResult = 0.0;
58      double result = instance.subtract(number1, number2);
59      assertEquals(expResult, result, 0.0);
60  }
61
62  /**
63   * Test of multiply method, of class Calculando.
64   */
65  @Test
```



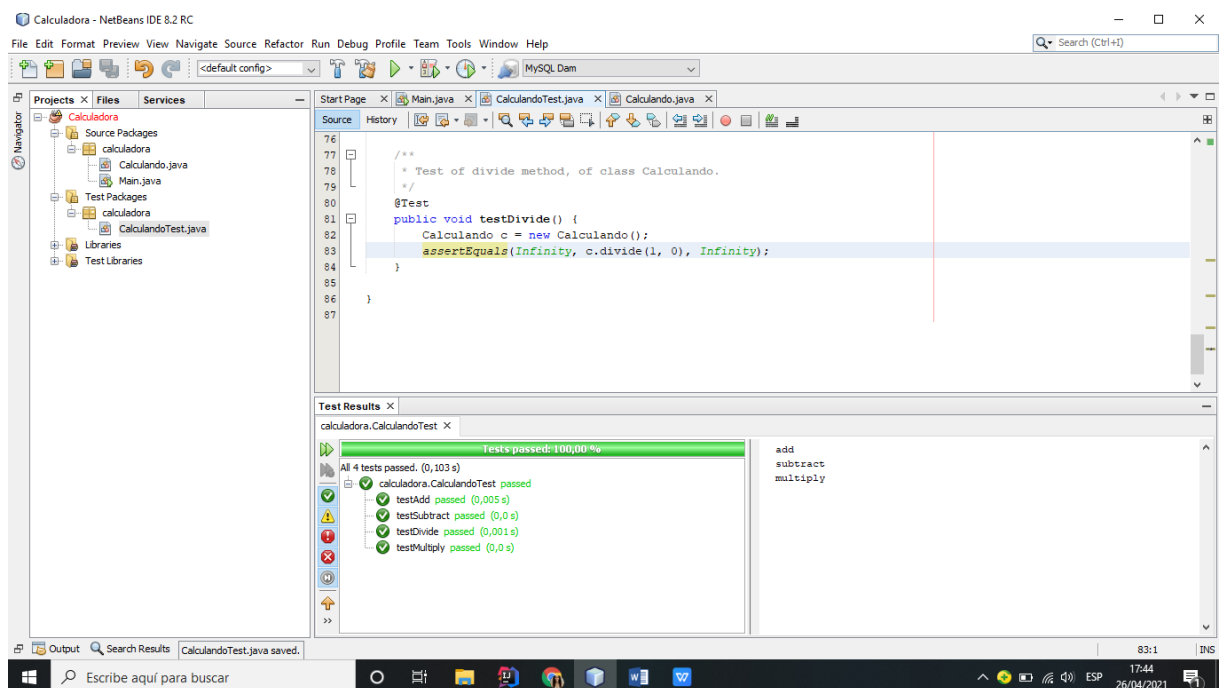
```
63  * Test of multiply method, of class Calculando.
64  */
65  @Test
66  public void testMultiply() {
67      System.out.println("multiply");
68      double number1 = 0.0;
69      double number2 = 0.0;
70      Calculando instance = new Calculando();
71      double expectedResult = 0.0;
72      double result = instance.multiply(number1, number2);
73      assertEquals(expResult, result, 0.0);
74  }
75
76  /**
77   * Test of divide method, of class Calculando.
78   */
79  @Test
80  public void testDivide() {
81      System.out.println("divide");
82      double number1 = 0.0;
83      double number2 = 0.0;
84      Calculando instance = new Calculando();
85      double expectedResult = 0.0;
86      double result = instance.divide(number1, number2);
87      assertEquals(expResult, result, 0.0);
88  }
89
90  }
91  }
```

Ejercicio 4

Selecciona la clase de prueba y ejecútala de nuevo. Debes de corregir todos los errores asignándole valores a las variables. Al final, debes de conseguir que la ejecución de la prueba sea satisfactoria. Como solución a este apartado deberás de aportar el código de la clase de prueba una vez que ha sido modificado para conseguir que las pruebas fueran satisfactorias. (Calificación 1 punto)

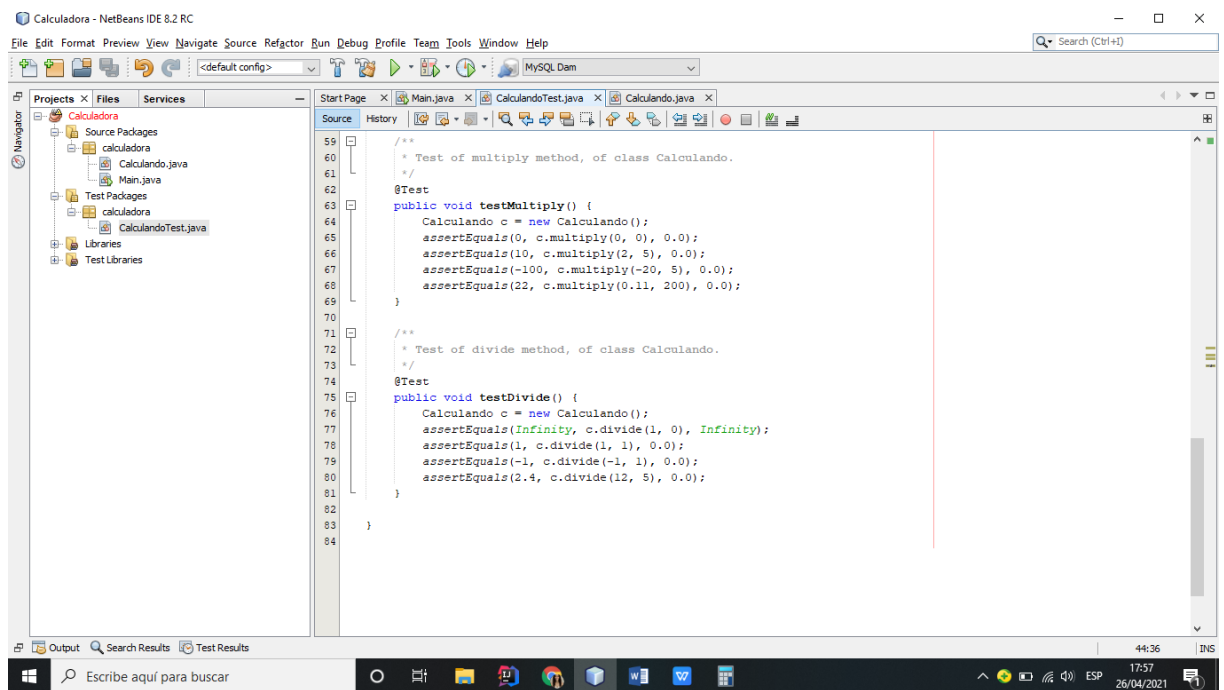
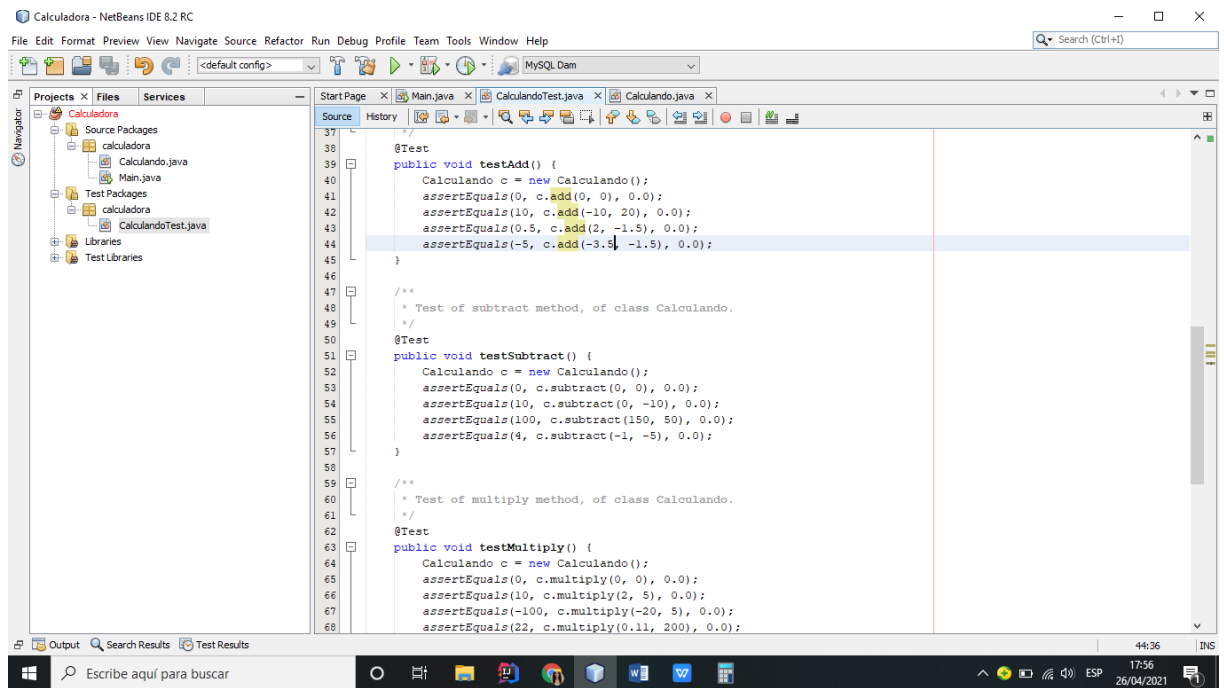


Corregimos el fallo que había en el test de dividir ya que al ser los dos números 0 salía NaN por lo que cambiamos el primer número por 1 y el resultado esperado en este caso sería infinito

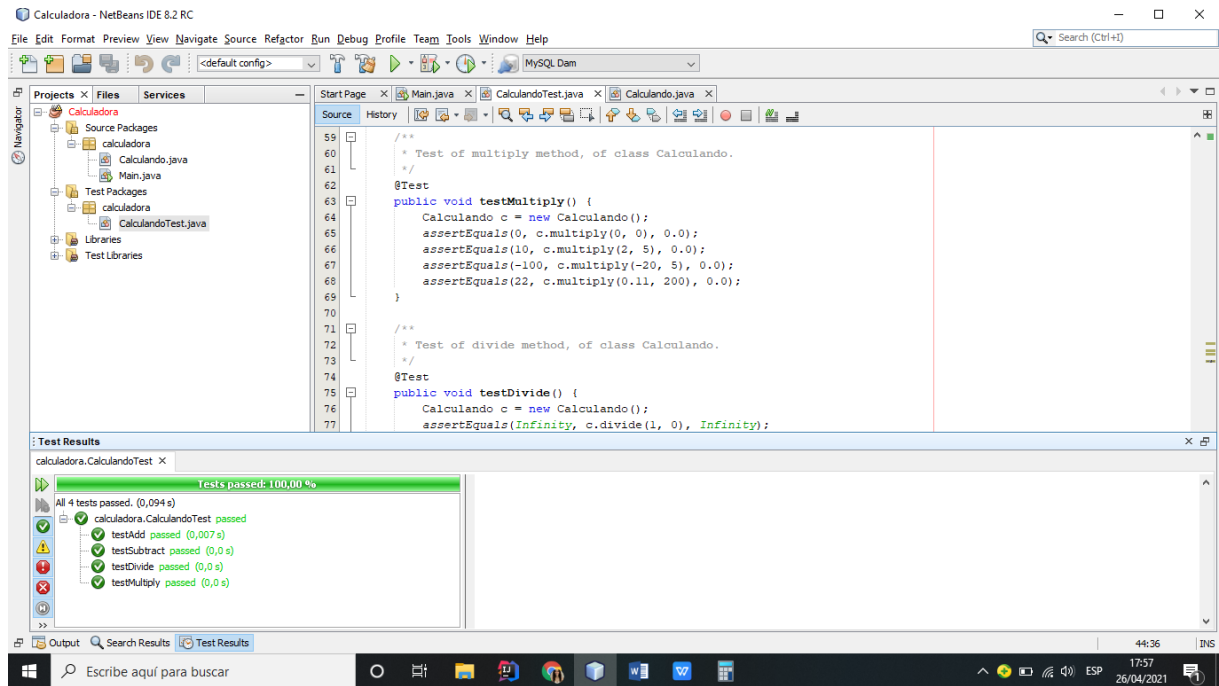


Añadimos más test

Modificamos los test y añadimos más pruebas en cada método con valores extremos que podrían hacer que fallase



Comprobamos el resultado de los test



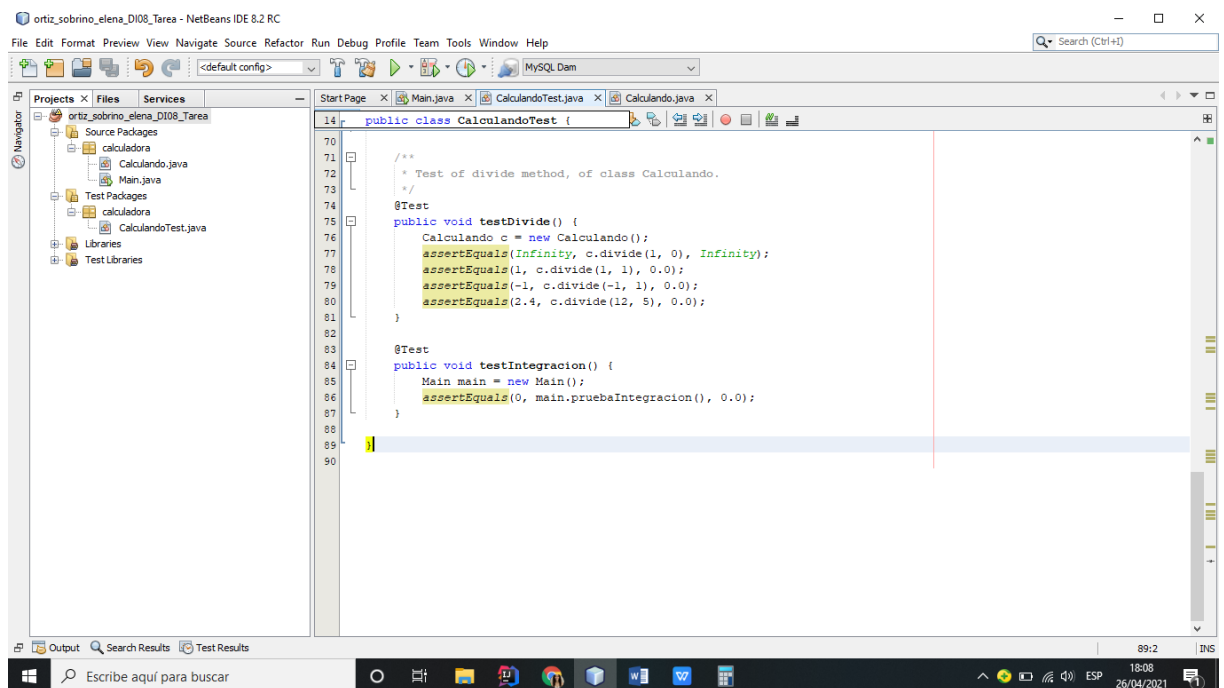
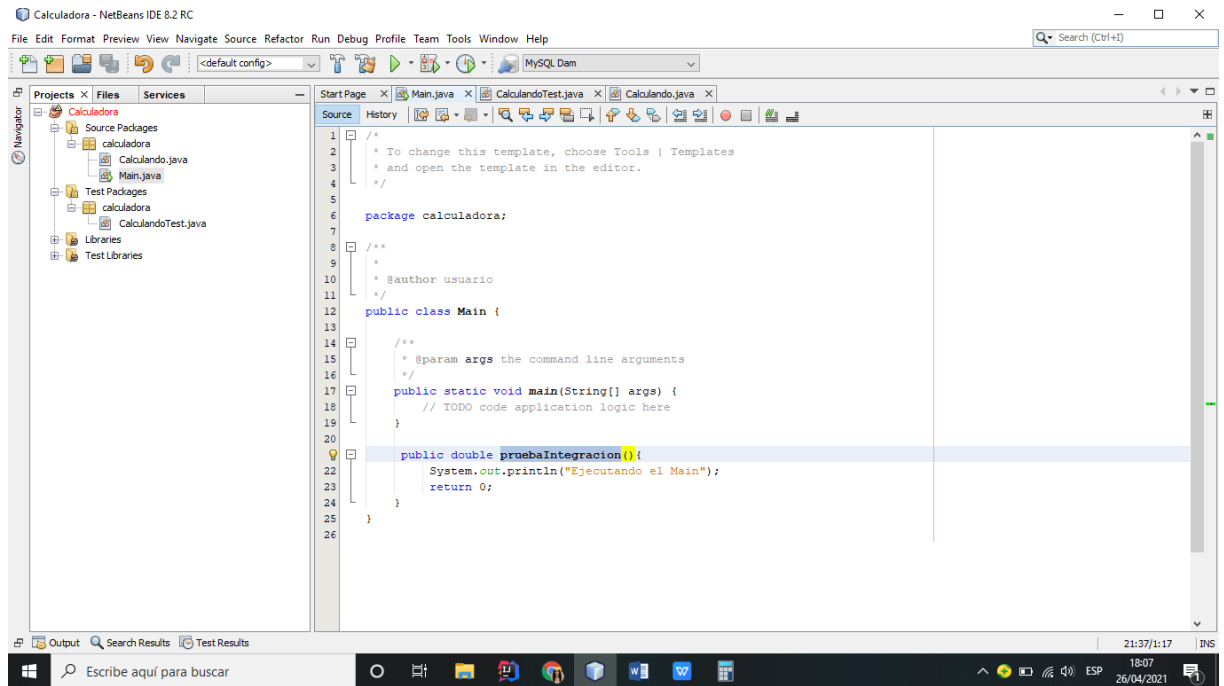
Ejercicio 5

Implementa la planificación de las pruebas de integración, sistema y regresión. (Calificación 2 puntos)

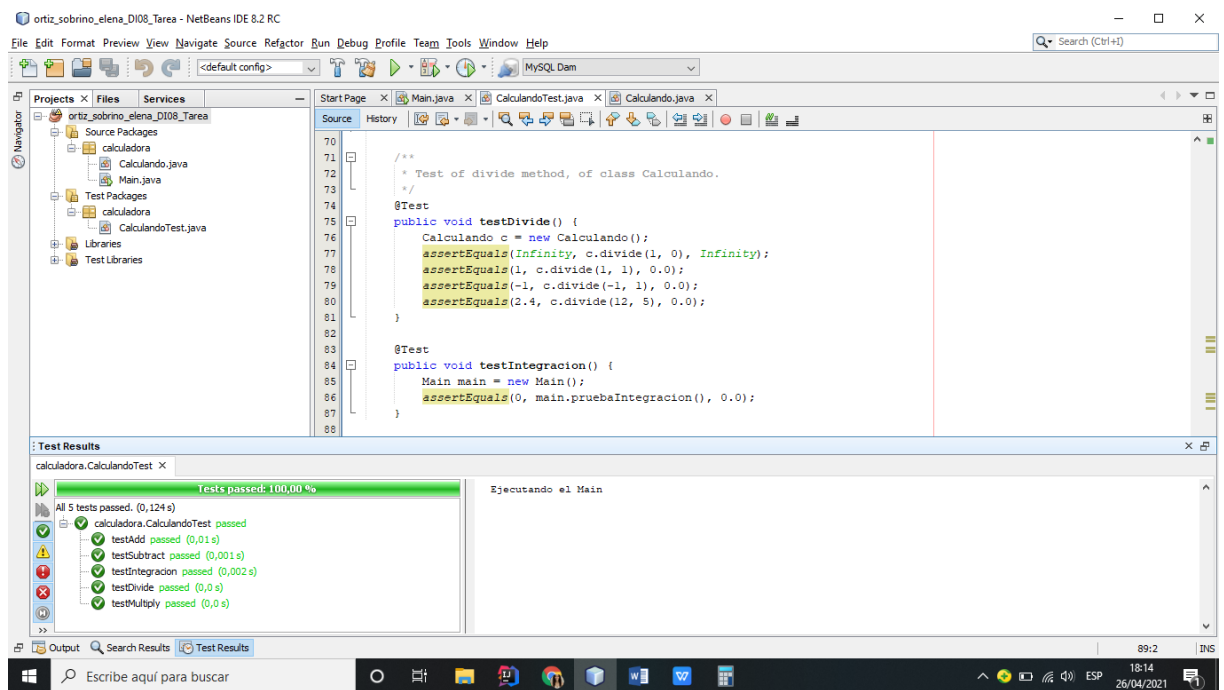
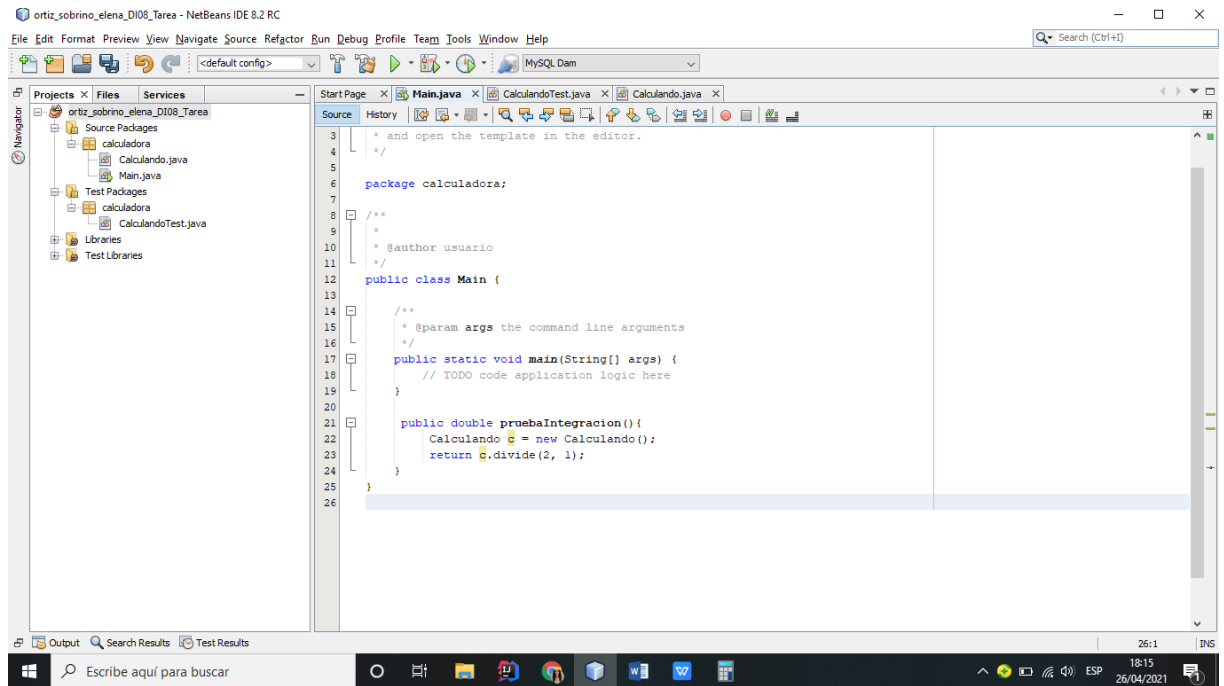
Integración.

Para realizar las pruebas de integración crearemos un método dentro de esta clase que instancie a la clase Calculando y use uno de sus métodos, para posteriormente hacer un test comprobando que el resultado es el esperado

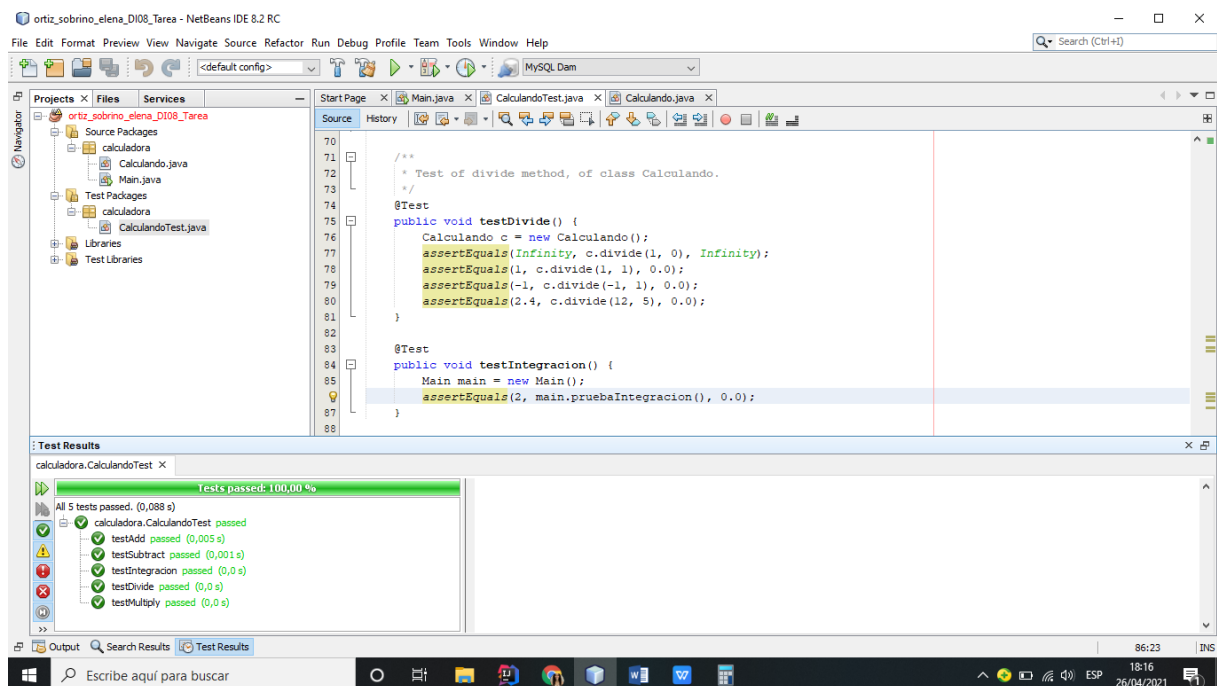
Creamos el método para la prueba de integración y comprobamos que funciona



Vemos que imprime por pantalla que se ha ejecutado el main y el resultado es el esperado. Ahora modificaremos el método `pruebaIntegracion()` del main y e instanciaremos la clase `Calculando` y llamaremos al método `divide()`



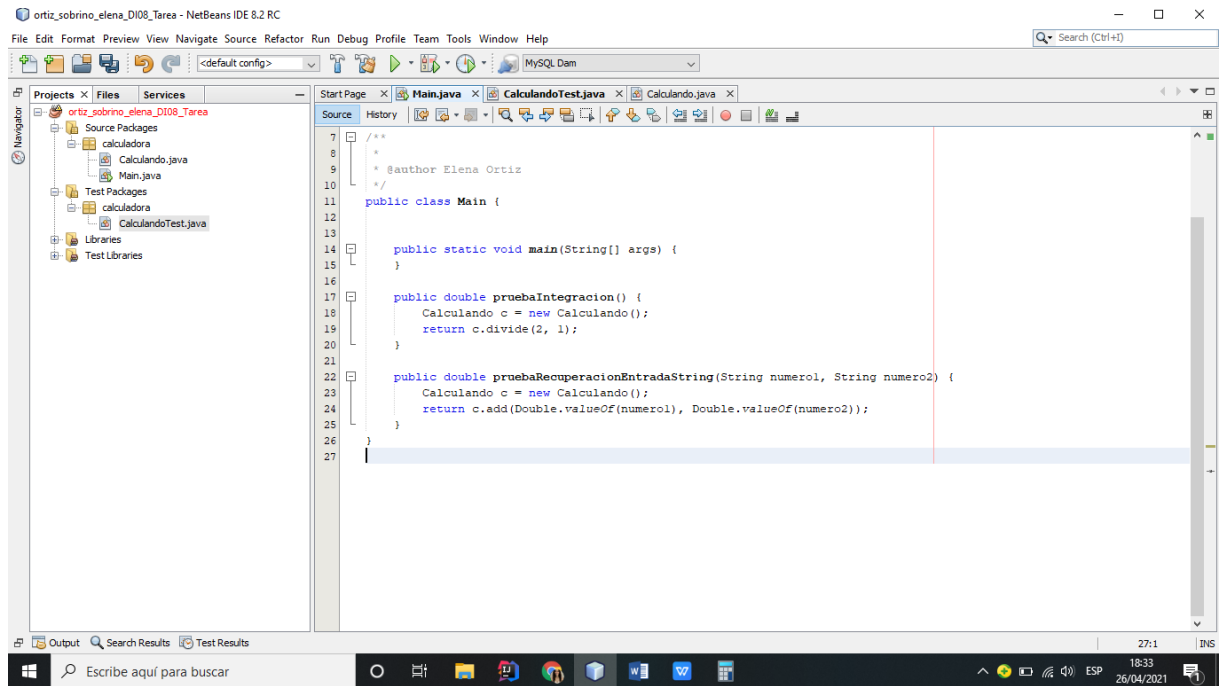
Modificamos el test y comprobamos que el `testIntegracion()` devuelve el resultado esperado



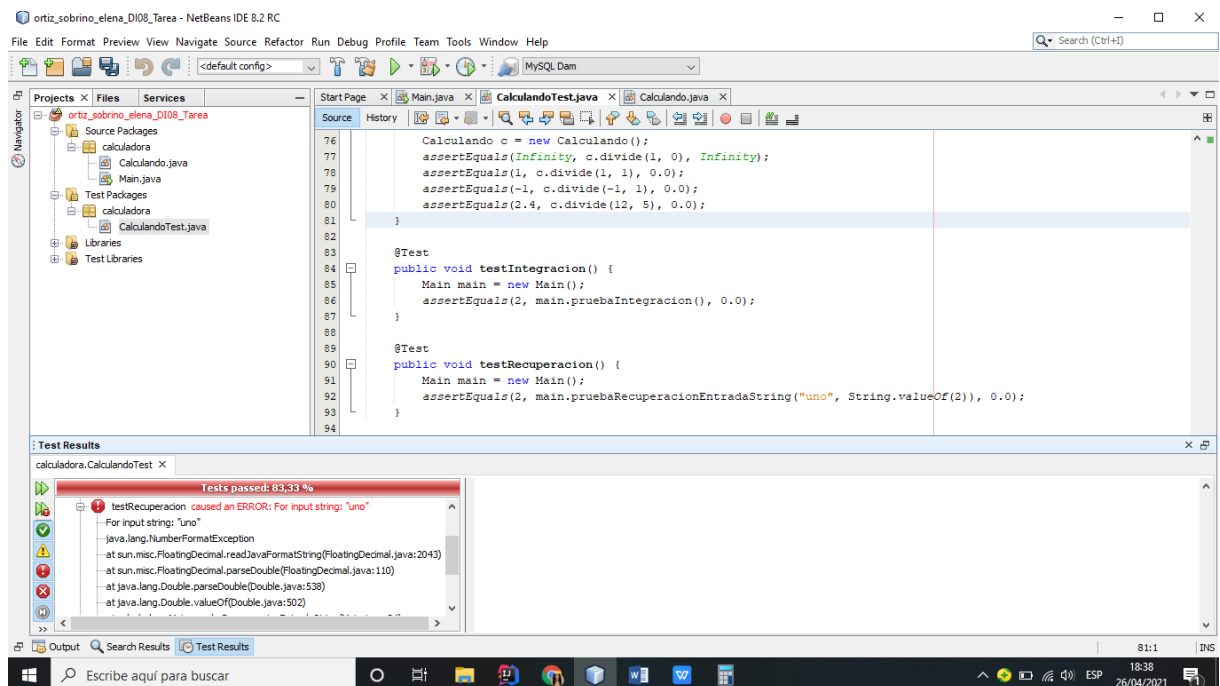
Sistema

Este tipo de pruebas verifican que el funcionamiento de programa en su conjunto cumple los requisitos de seguridad, fiabilidad, exactitud y velocidad. Con las pruebas unitarias hemos comprobado que la fiabilidad y exactitud de los cálculos de nuestro programa es correcta, además podemos ver que el tiempo para ejecutar los test ha sido rápido por lo que podemos considerar que nuestro programa trabaja de forma rápida.

Otra de las pruebas de sistema que podemos realizar es la de Recuperación forzando el fallo del software para comprobar si es capaz de recuperarse de forma satisfactoria. Estas pruebas las realizaremos introduciendo entradas no esperadas por la aplicación



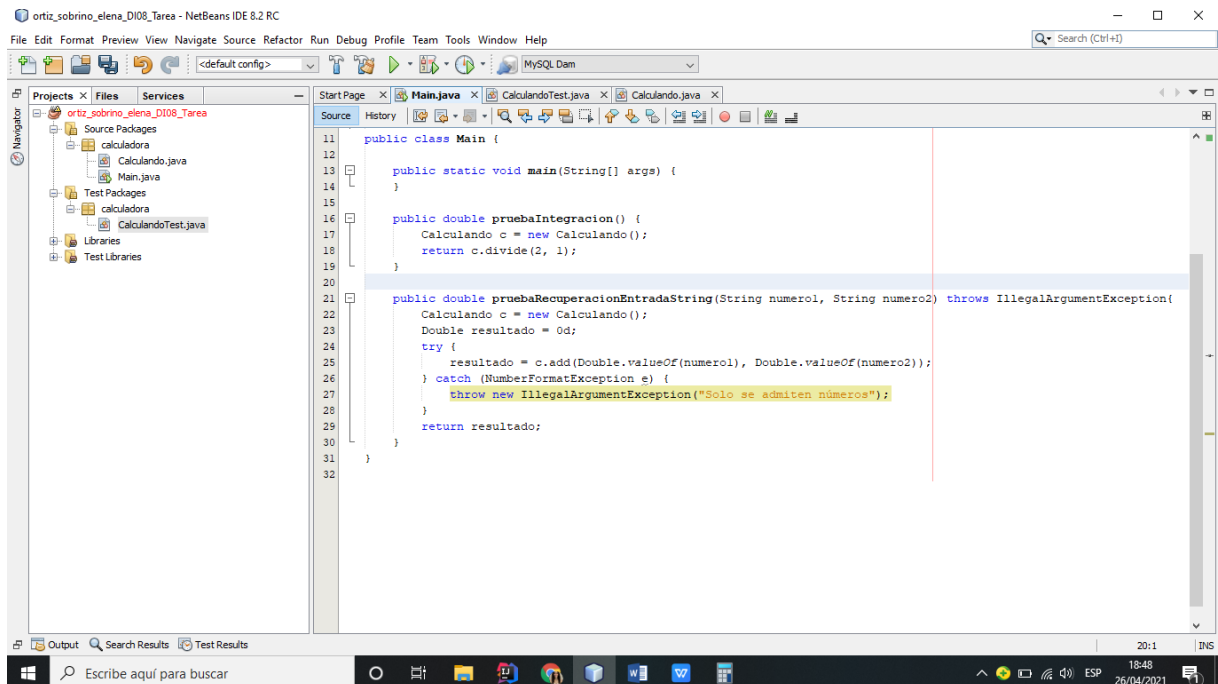
En este caso fallaría porque no puede convertir el texto "uno" a número por lo que habría que validar las entradas de datos del usuario previamente



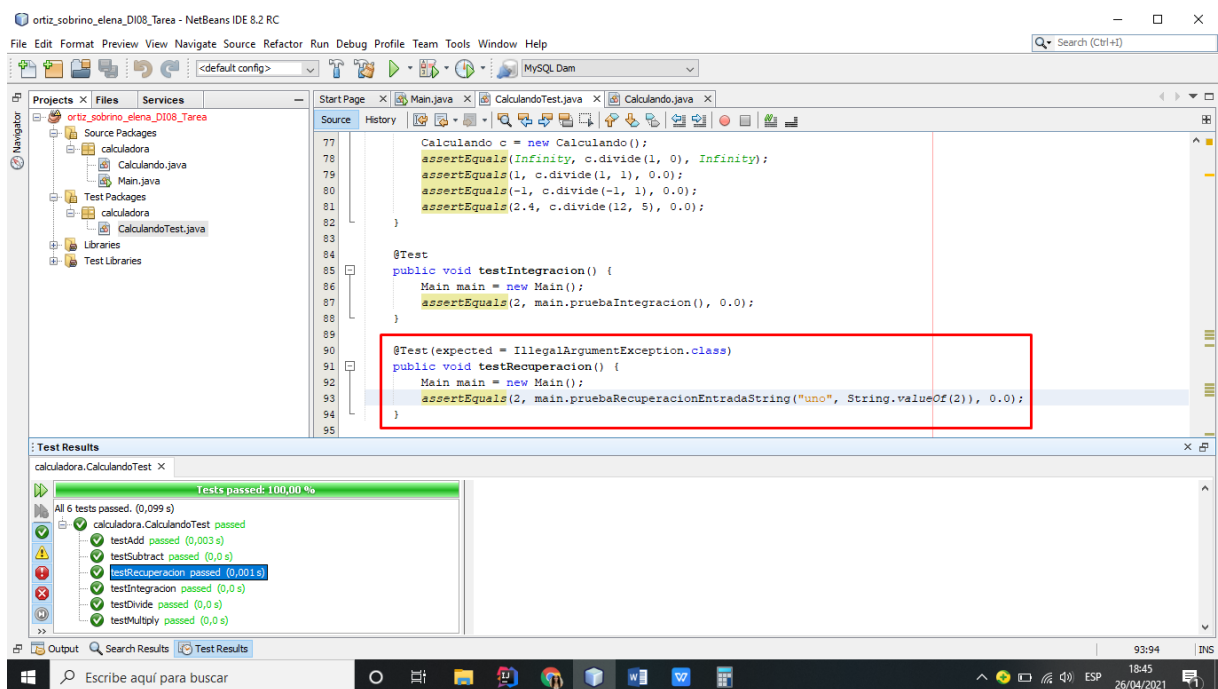
Para solucionar esto habría que controlar las entradas del usuario dentro del método pruebaRecuperacionEntradaString() del Main.

Lo que haré será que cuando el programa lance una excepción NumberFormatException capturarla y transformarla en una excepción IllegalArgumentException con el mensaje "Solo se admiten números"

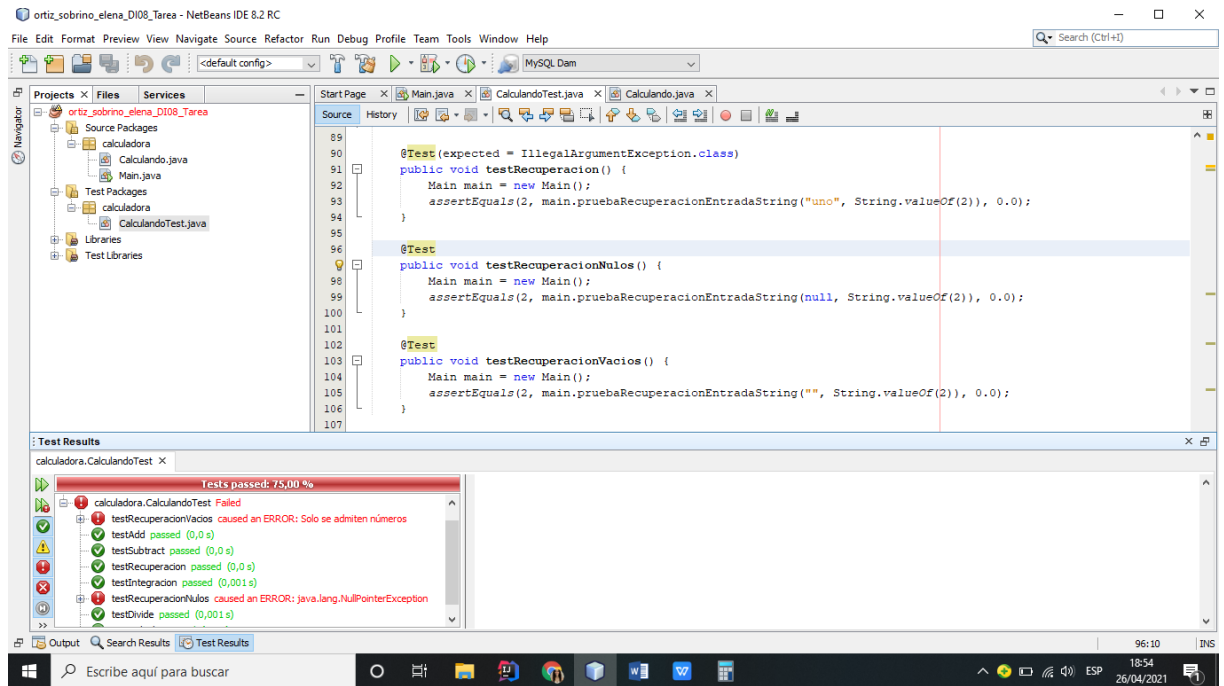
para que en caso de fallo el usuario tenga más información del error y la persona que use este método tenga que controlar la excepción



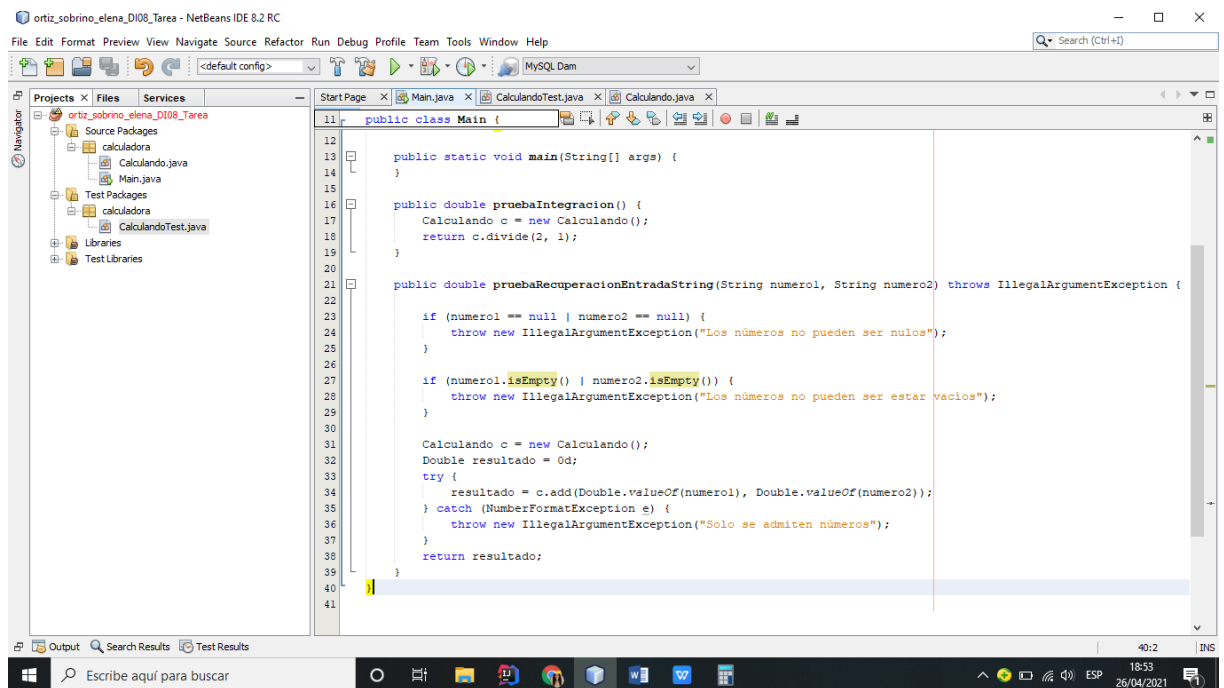
En los test comprobamos que para una entrada no válida lanza excepción

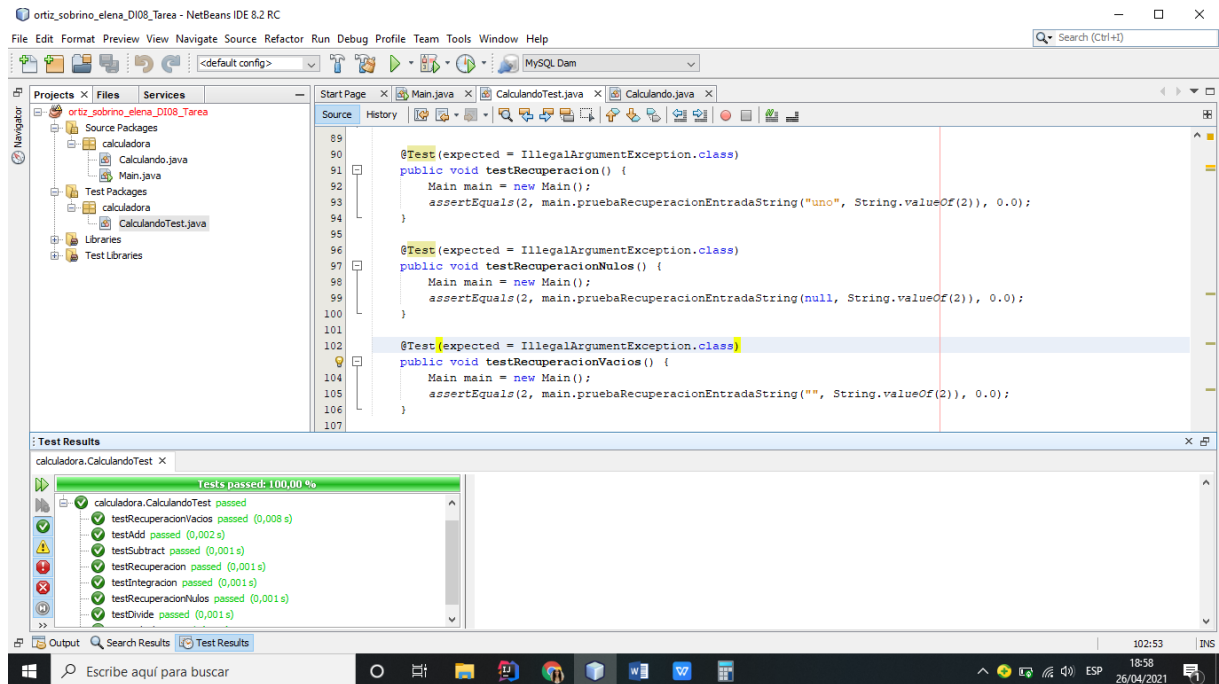


Comprobamos qué ocurriría si ponemos uno de los números nulos o vacíos y vemos que lanza NullPointerException y lanza el mensaje de que solo admite números cuando el campo en realidad está vacío



Para solucionarlo controlamos las excepciones e indicamos a los test que se espera la excepción `IllegalArgumentException` en ambos casos





Regresión

Las pruebas de regresión se realizan cuando hemos realizado modificaciones en el programa puesto que podemos introducir algún fallo en los cambios por eso es necesario volver a ejecutar las pruebas que se han realizado anteriormente para asegurarse de que no hemos introducido cambios no deseados durante las modificaciones.

Estas pruebas se han ejecutado indirectamente cuando se hacían las pruebas anteriores cada vez que ejecutábamos los test de junit en los que se probaba las diferentes partes del programa

Ejercicio 6

Planifica las restantes pruebas, estableciendo qué parámetros se van a analizar. (Calificación 2 puntos)

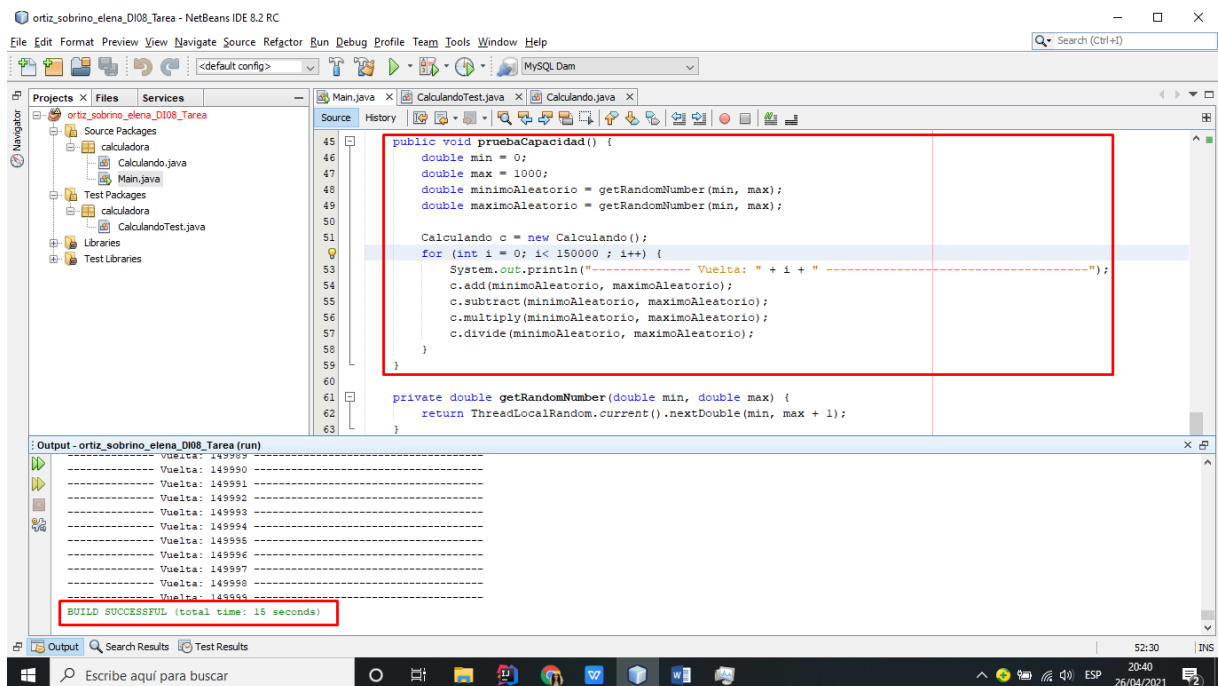
De aceptación

Estas pruebas validan si el comportamiento del software cumple con sus características. Como en este caso las funciones que debe realizar el software es devolver el resultado correcto de operaciones aritméticas (suma, resta, multiplicación y división) estas pruebas se han realizado al hacer las pruebas unitarias usando cada método con diferentes valores y comprobando que el resultado es el esperado

De capacidad

Esta prueba nos permite ver cómo se comporta el programa en condiciones extremas. En este caso un ejemplo de condiciones extremas sería la realización de operaciones matemáticas con una gran cantidad de números por lo que crearemos un test donde dentro de un bucle realizaremos diferentes operaciones matemáticas de la clase Calculando.

Aquí vemos que realiza 150.000 operaciones matemáticas en 15 segundos lo que sería unos 10.000 por segundo lo que significa que el programa tiene la capacidad de realizar un alto número de operaciones sin que el programa se pare



De rendimiento

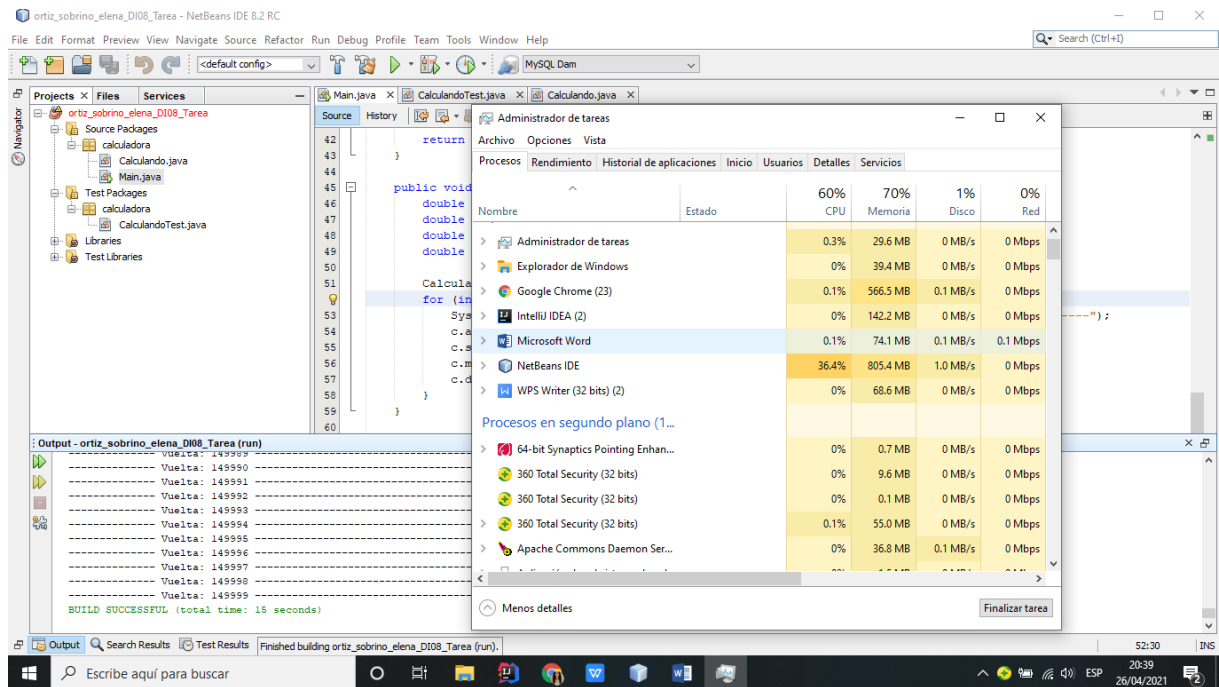
La prueba de rendimiento consiste en determinar los tiempos de respuesta de la aplicación y se ha realizado con la prueba de capacidad donde se realizaban 150.000 operaciones en 15 segundos que serían 10.000 por segundo por lo que el rendimiento del programa es aceptable

De uso de recursos

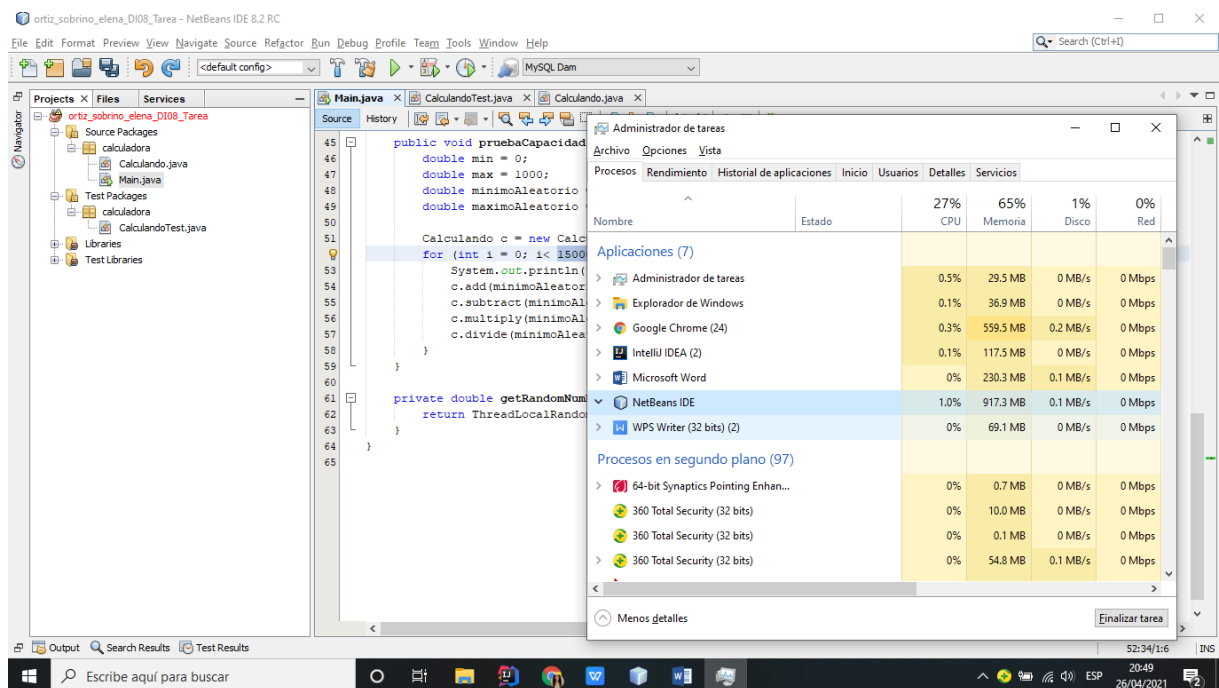
Esta prueba se puede realizar monitorizando el sistema durante las pruebas de capacidad y rendimiento donde observaremos el aumento del consumo de recursos del ordenador durante la ejecución del programa

Aquí vemos que el consumo de la CPU aumenta durante la ejecución del programa, sin embargo, la memoria RAM no aumenta durante esta prueba por lo que la conclusión que se puede extraer es que este programa aumenta el uso de recursos del ordenador cuando se usa a pleno rendimiento, pero consumiendo como máximo una tercera parte de la CPU por lo que podría decirse que realiza un uso eficiente de recursos

(Captura durante la ejecución del programa)



(Captura al finalizar el programa)



Funcionales

Estas pruebas se han realizado junto con los test unitarios al comprobar los valores límite como, por ejemplo, el valor infinito en la división o los valores extremos comprobando así que el programa devuelve los valores deseados hasta en casos extremos.

Ejercicio 7

Supuestas exitosas las pruebas, documenta el resultado (Calificación 2 puntos).

Esta documentación se ha hecho durante la realización de los ejercicios anteriores.