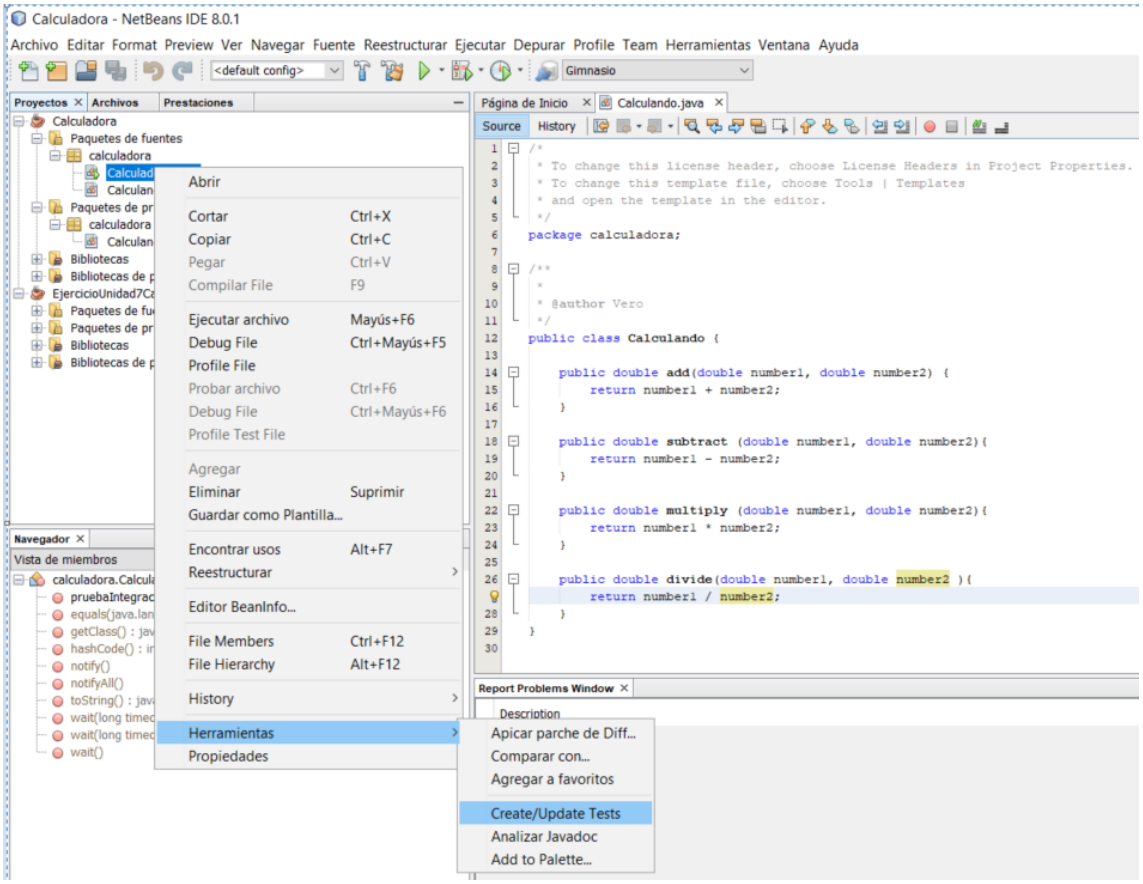
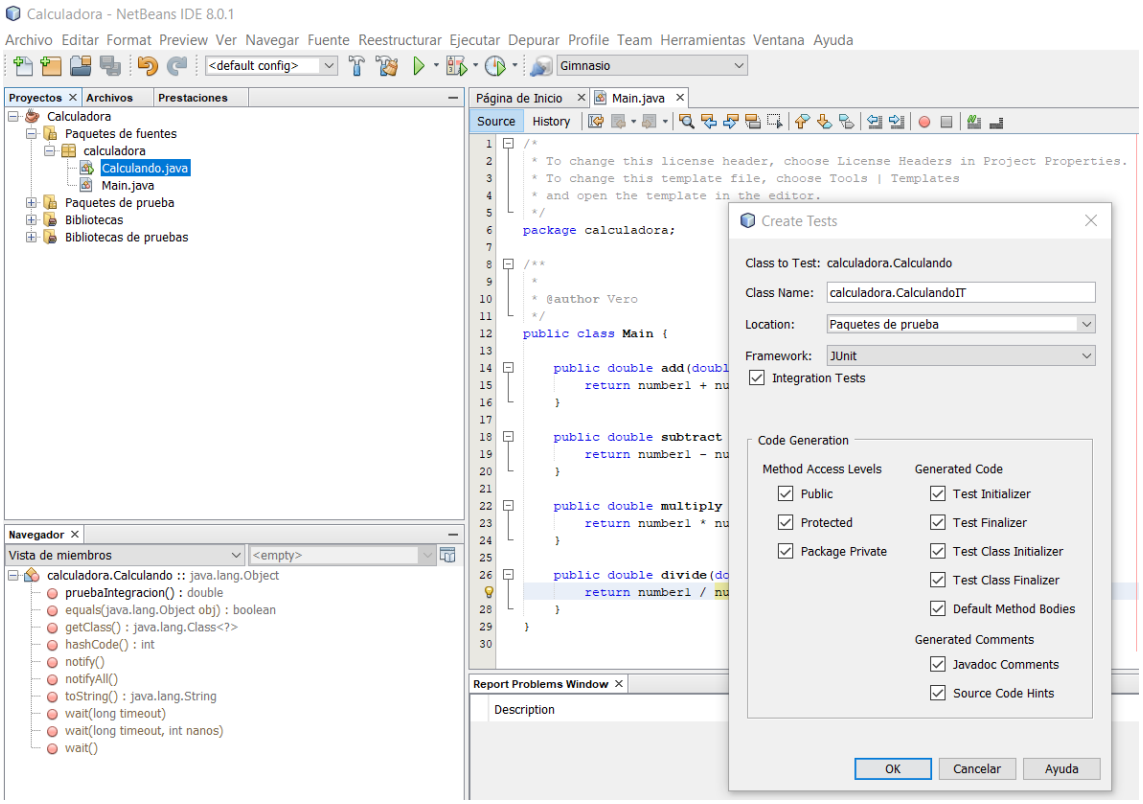
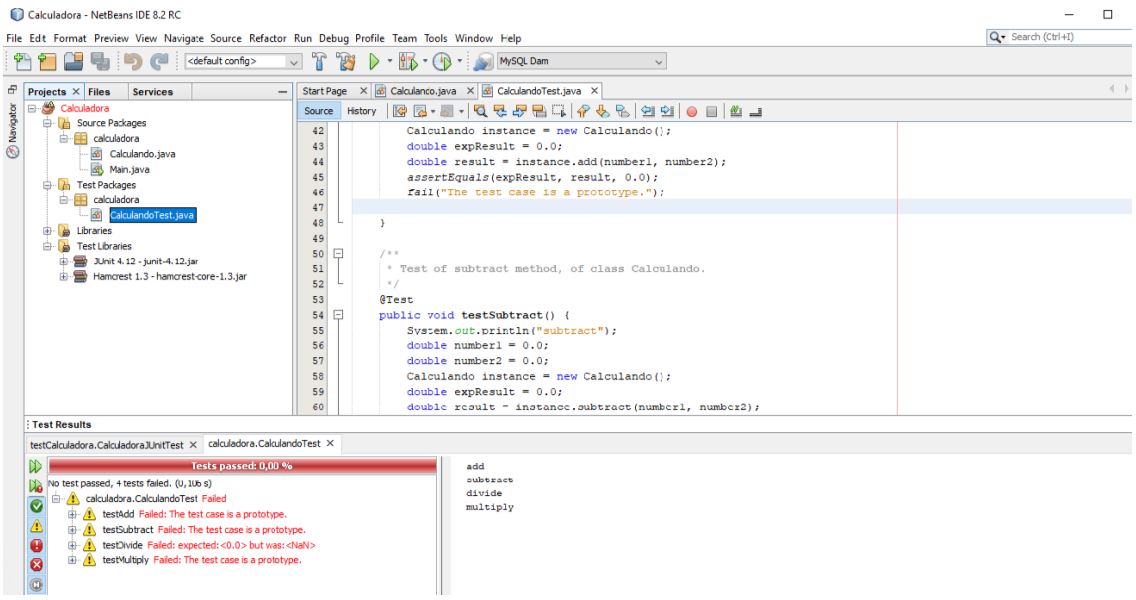
Ejemplo calculadora. Hemos definido una serie de métodos en la clase Calculando.java (suma, resta, división y multiplicación). Vamos a probar cada método de la clase con JUnit. Para ello, deberás de seleccionar la clase y en el menú Herramientas deberás de seleccionar la opción Create /update Tests.
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.



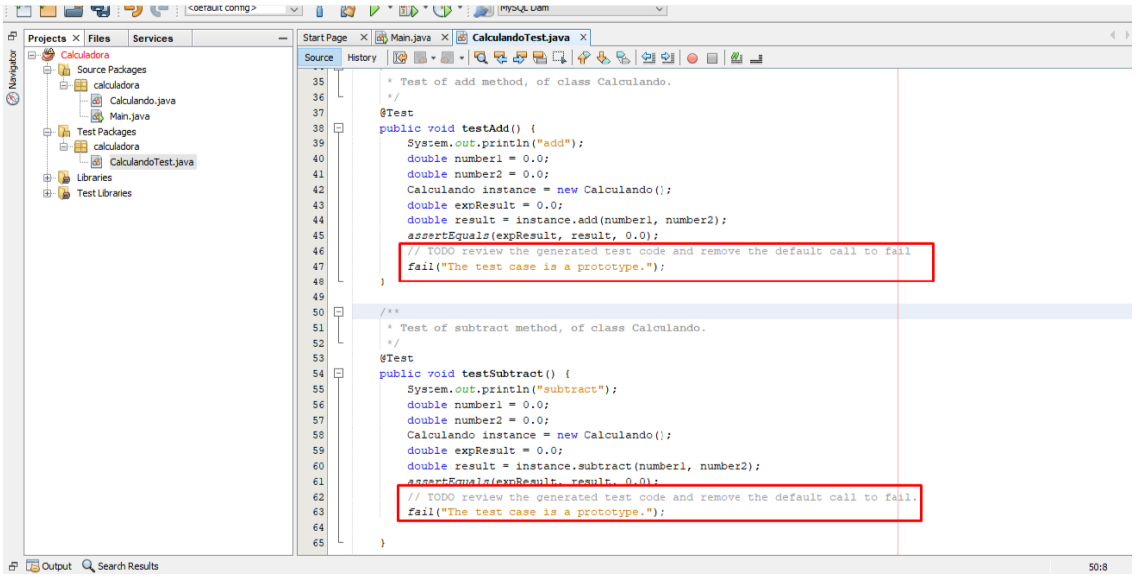


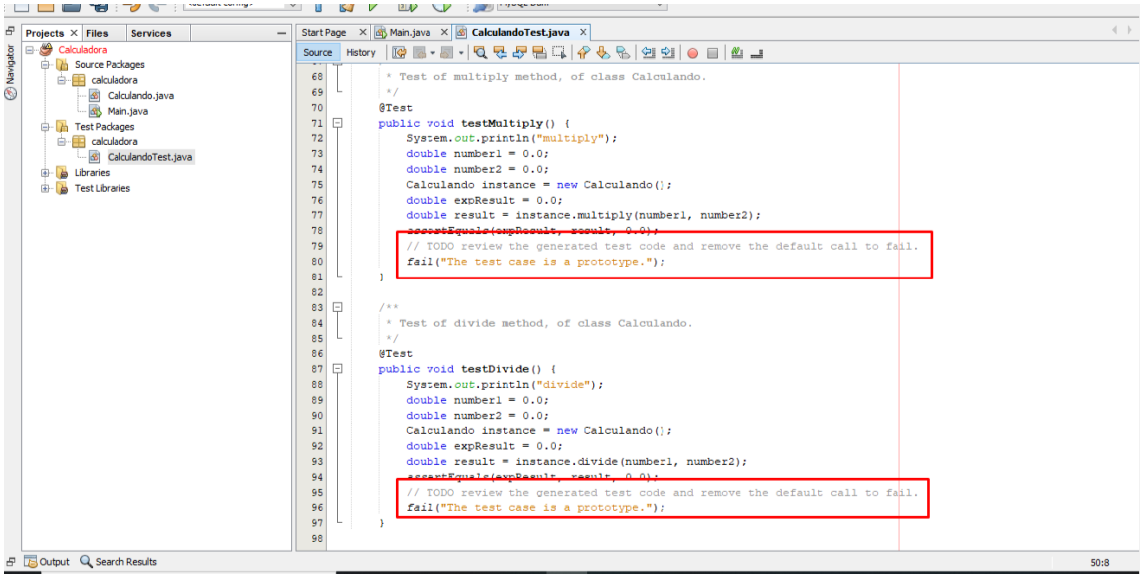
Vemos que los 4 test que se han creado automáticamente con la clase pruebas fallan

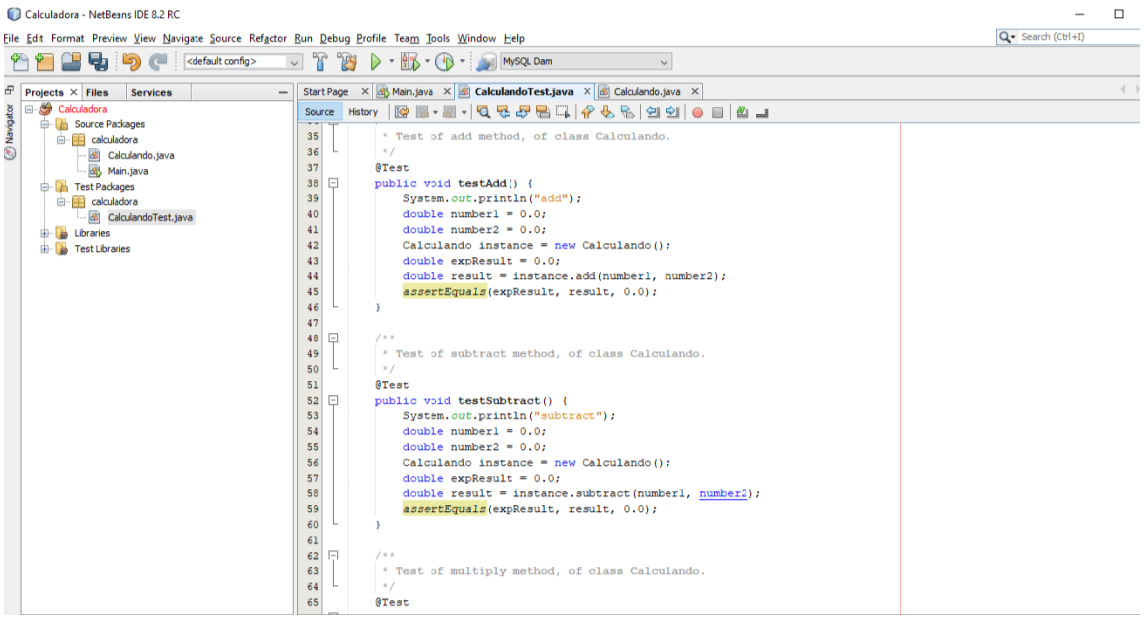


Accede al código de la clase de pruebas y elimina las líneas:
// TODO review the generated test code and remove the default call to fail.
fail("The test case is a prototype.");que aparece al final de cada método.

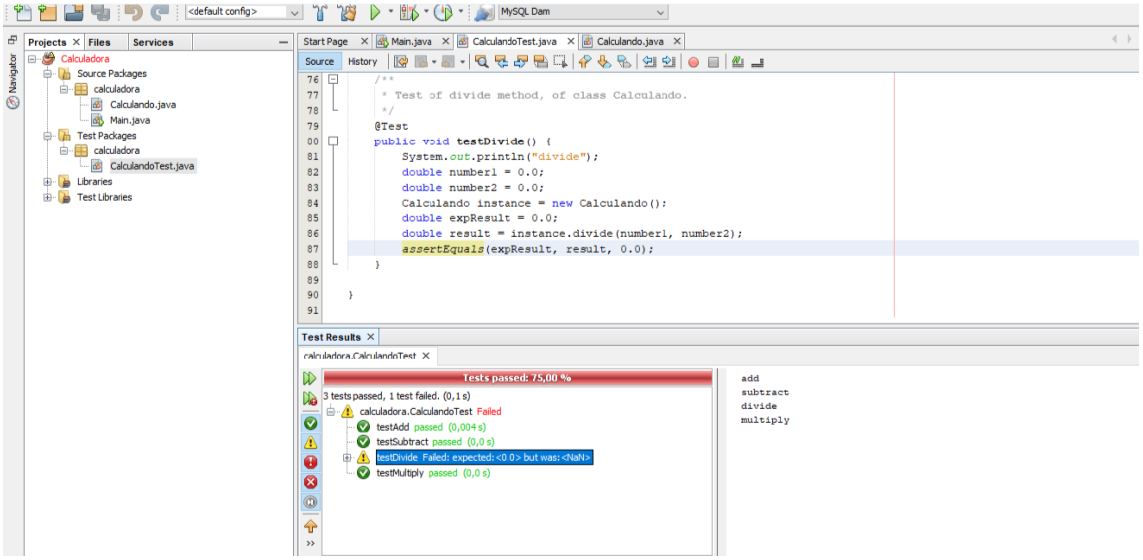
Netbeans autogenera el código de test de la clase Calculando que incluye los métodos sumar, restar, multiplicar y dividir.


Projects
Files
Services
Calculadora
Source Packages
calculadora
Calculando.java
Main.java
Test Packages
calculadora
CalculandoTest.java
Libraries
Test Libraries
Start Page
Main.java
CalculandoTest.java
Source
History
* Test of add method, of class Calculando.
@Test
public void testAdd() {
System.out.println("add");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.add(number1, number2);
assertEquals(expResult, result, 0.0);
// TODO review the generated test code and remove the default call to fail
fail("The test case is a prototype.");
* Test of subtract method, of class Calculando.
@Test
public void testSubtract() {
System.out.println("subtract");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.subtract(number1, number2);
// TODO review the generated test code and remove the default call to fail.
fail("The test case is a prototype.");
Output
Search Results
50:8

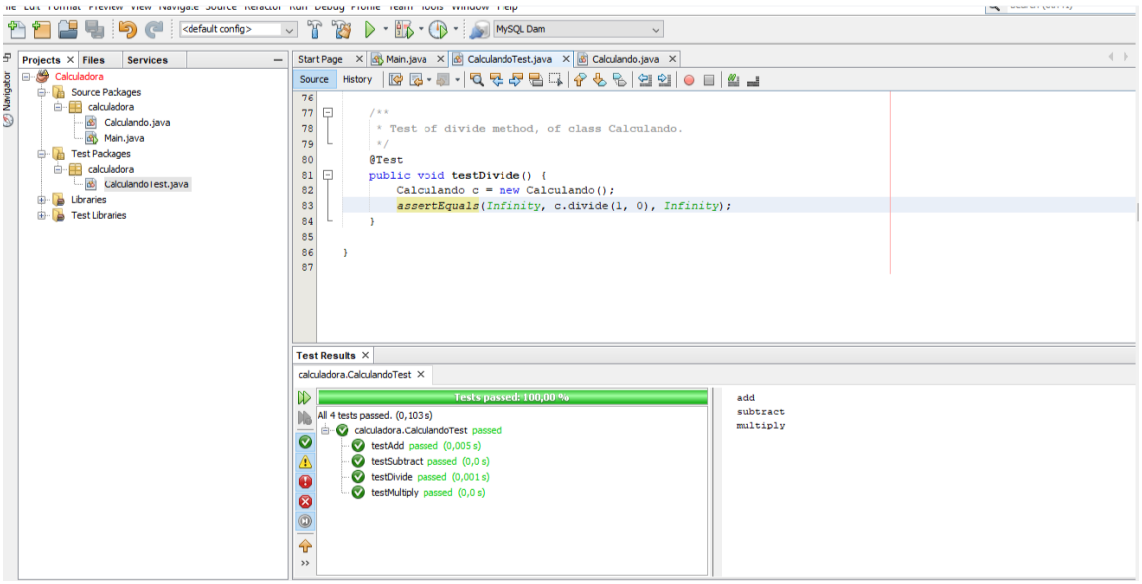

Projects
Files
Services
Calculadora
Source Packages
calculadora
Calculando.java
Main.java
Test Packages
calculadora
CalculandoTest.java
Libraries
Test Libraries
Start Page
Main.java
CalculandoTest.java
Source
History
* Test of multiply method, of class Calculando.
@Test
public void testMultiply() {
System.out.println("multiply");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.multiply(number1, number2);
// TODO review the generated test code and remove the default call to fail.
fail("The test case is a prototype.");
* Test of divide method, of class Calculando.
@Test
public void testDivide() {
System.out.println("divide");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.divide(number1, number2);
// TODO review the generated test code and remove the default call to fail.
fail("The test case is a prototype.");
Output
Search Results
50:8


Calculadora - NetBeans IDE 8.2 RC
File Edit Format Preview View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
MySQL Dam
Projects
Files
Services
Calculadora
Source Packages
calculadora
Calculando.java
Main.java
Test Packages
calculadora
CalculandoTest.java
Libraries
Test Libraries
Start Page
Main.java
CalculandoTest.java
Calculando.java
Source
History
* Test of add method, of class Calculando.
@Test
public void testAdd() {
System.out.println("add");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.add(number1, number2);
assertEquals(expResult, result, 0.0);
* Test of subtract method, of class Calculando.
@Test
public void testSubtract() {
System.out.println("subtract");
double number1 = 0.0;
double number2 = 0.0;
Calculando instance = new Calculando();
double expResult = 0.0;
double result = instance.subtract(number1, number2);
assertEquals(expResult, result, 0.0);
* Test of multiply method, of class Calculando.
@Test

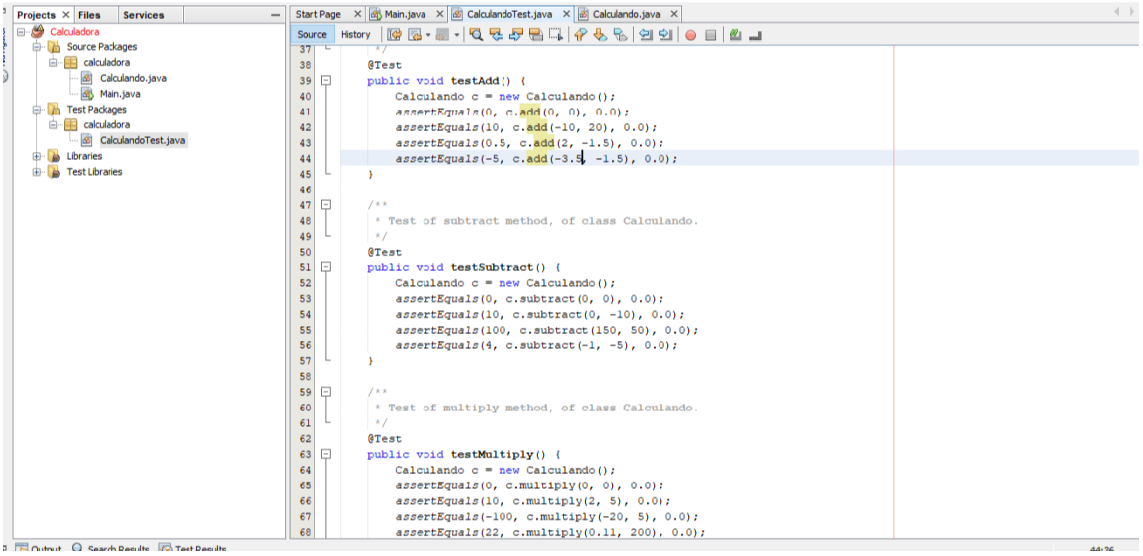
Selecciona la clase de prueba y ejecútala de nuevo. Debes de corregir todos los errores asignándole valores a las variables. Al final, debes de conseguir que la ejecución de la prueba sea satisfactoria.

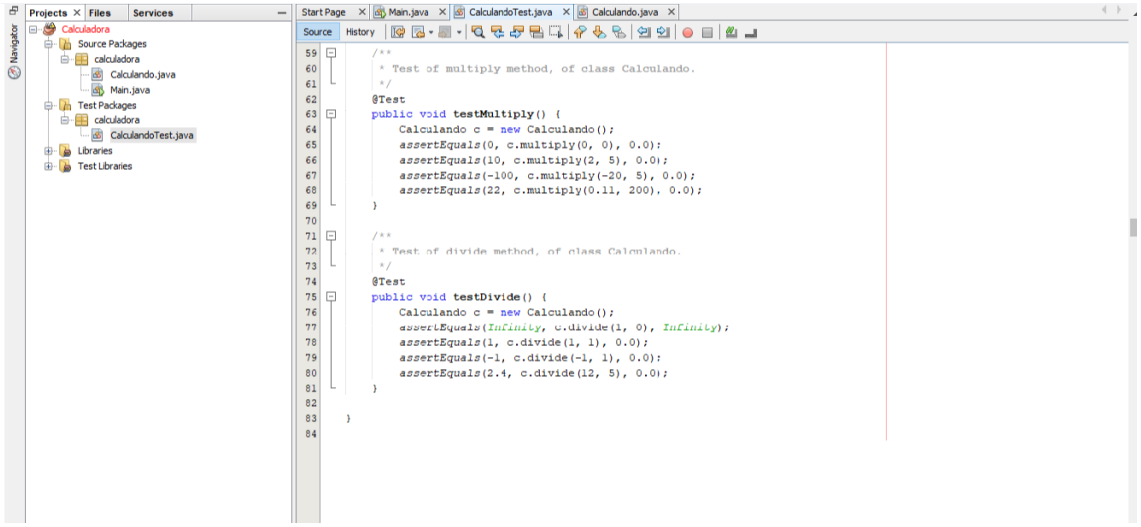


Corregimos el fallo que había en el test de dividir ya que al ser los dos números 0 salía NaN por lo que cambiamos el primer número por 1 y el resultado esperado en este caso sería infinito

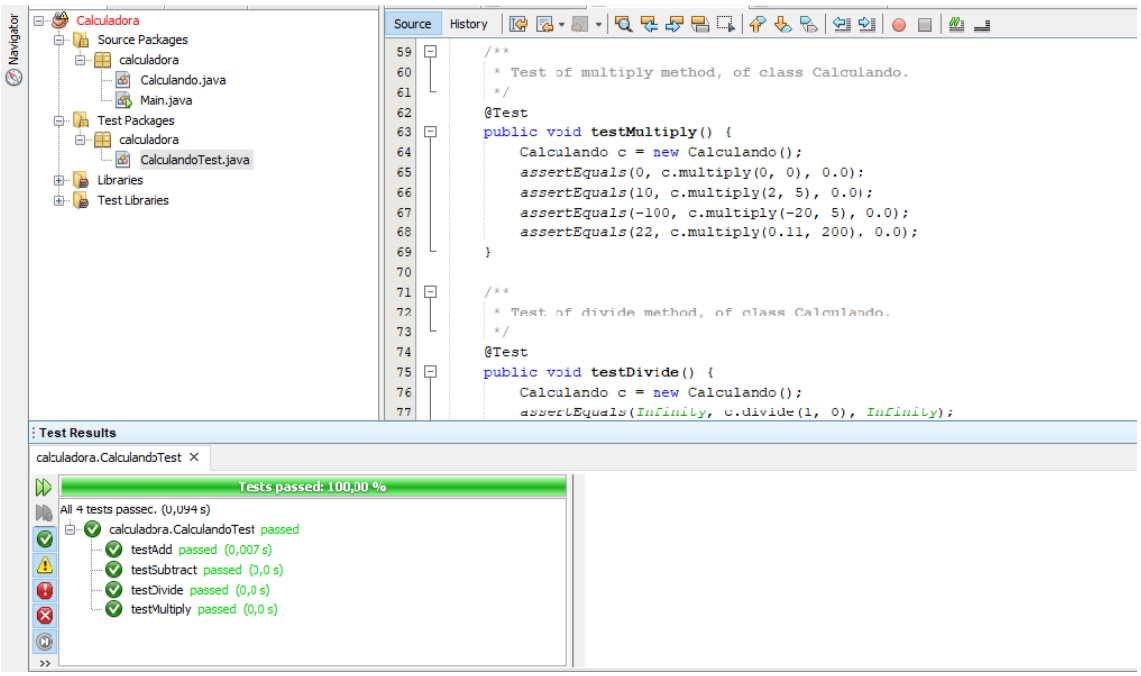


Modificamos los test y añadimos más pruebas en cada método con valores extremos que podrían hacer que fallase





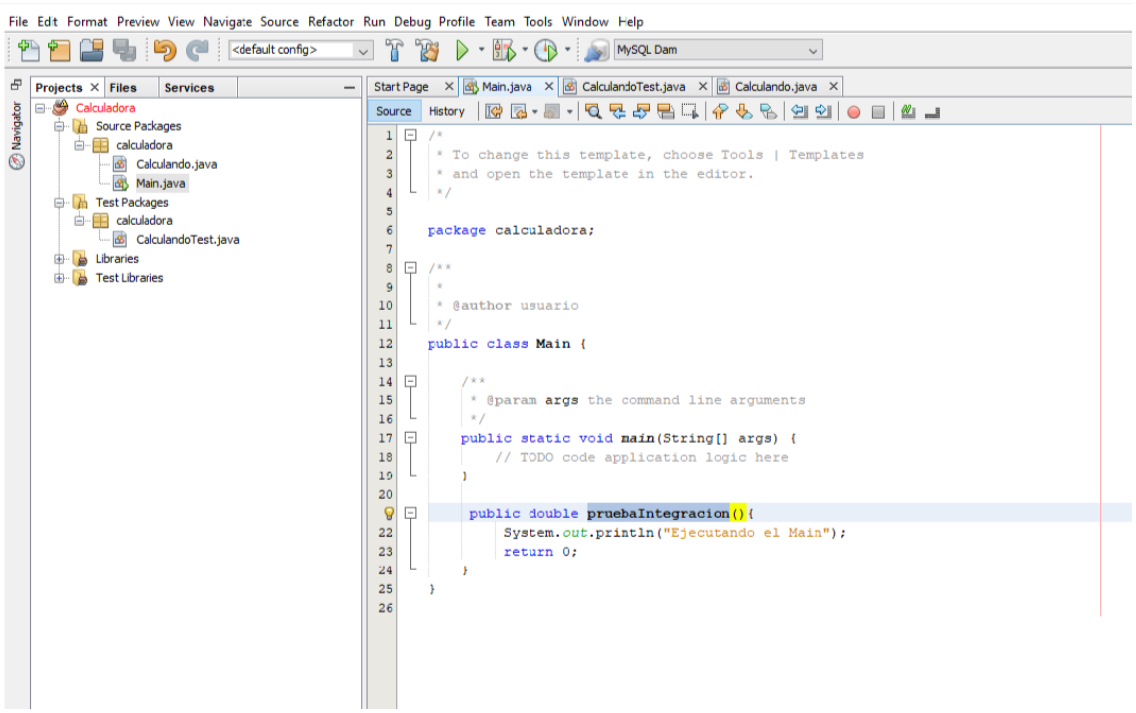
Comprobamos que funciona

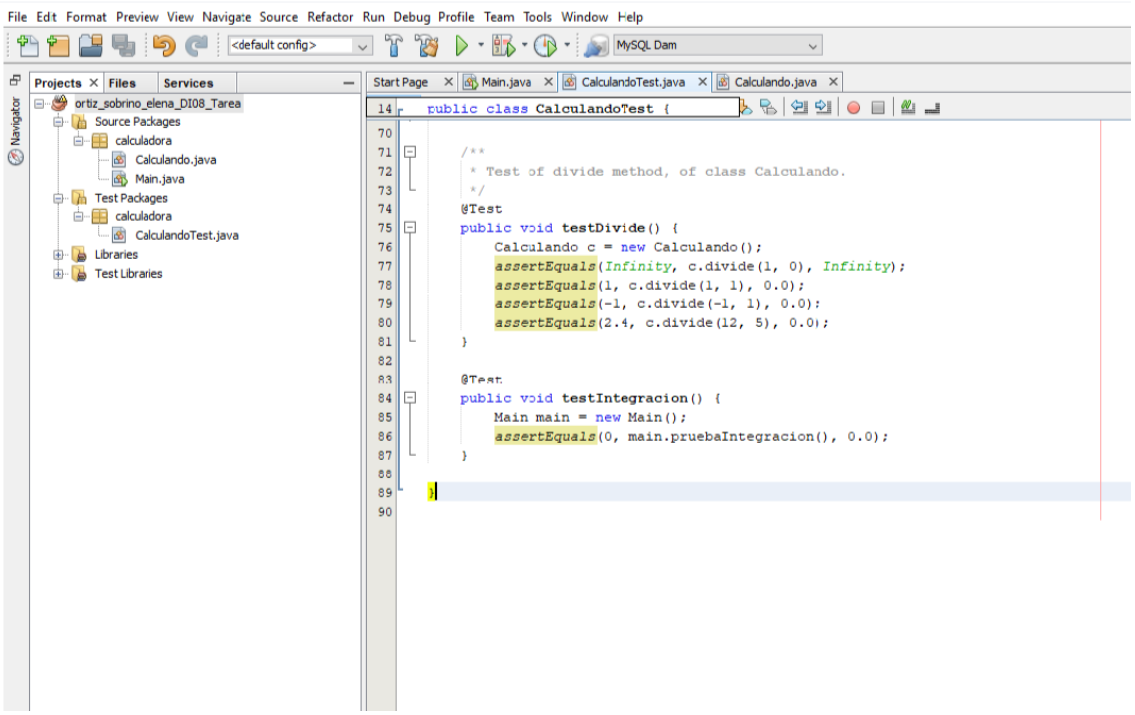


Implementa la planificación de las pruebas de integración

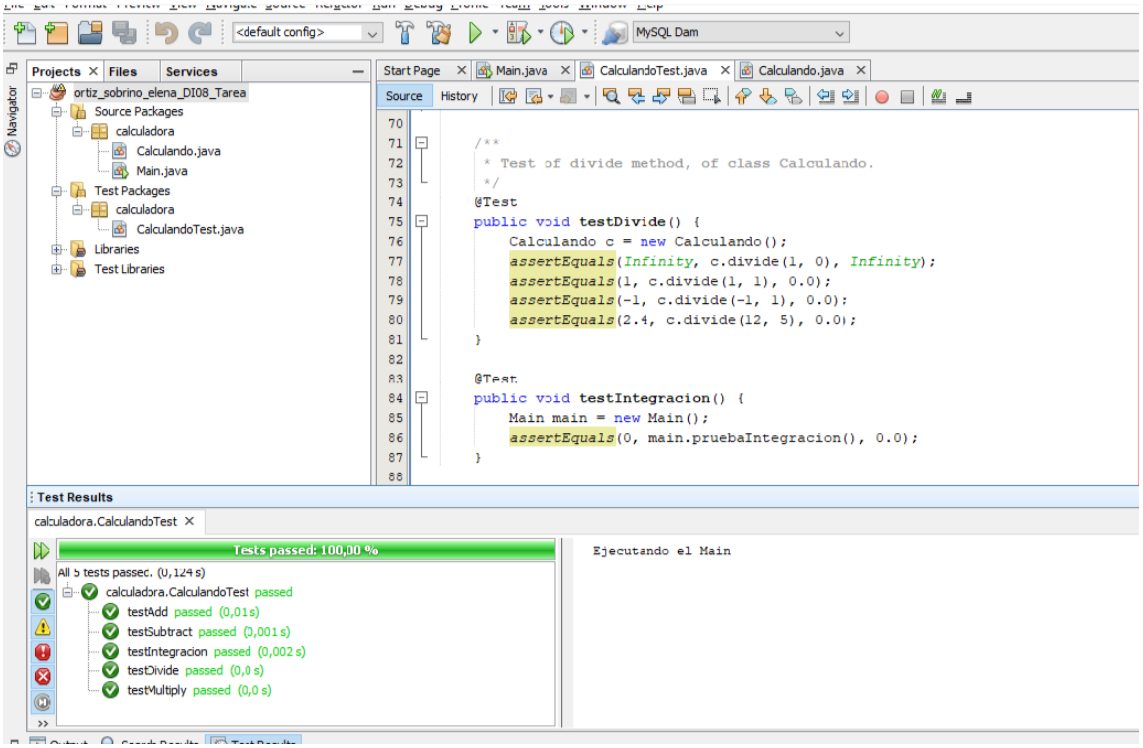
Integración.

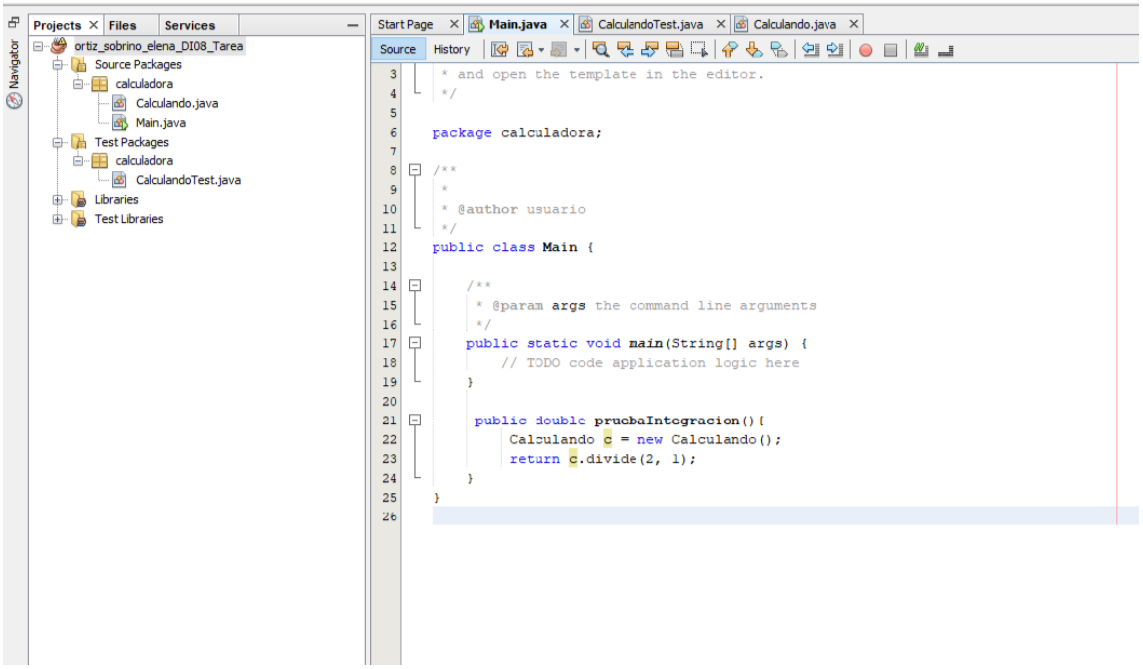
Para realizar las pruebas de integración crearemos un método dentro de esta clase que instancie a la clase Calculando y use uno de sus métodos, para posteriormente hacer un test comprobando que el resultado es el esperado .
Creamos el método para la prueba de integración y comprobamos que funciona





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