Ingeniería del Software: Laboratorio

Cuaderno de prácticas. P5 - Pruebas

Contenido

OBJETIVO

El presente cuaderno de prácticas debe servir como guía al alumno para la asimilación y estudio de los siguientes contenidos:

- Pruebas unitarias y de integración

RECOMENDACIÓN DE HERRAMIENTAS

- Netbeans (https://netbeans.org)

EJERCICIO TUTORIZADO – Pruebas con JUnit 5

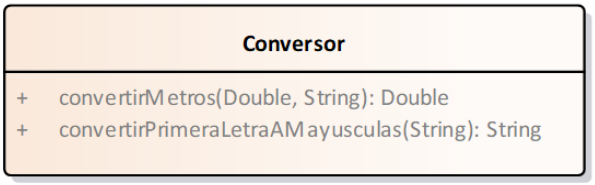
Enunciado

Se van a realizar pruebas unitarias de los dos métodos siguientes que ofrece una clase denominada “Conversor”:

- convertirMetros: recibe un primer parámetro con un valor en metros, y un segundo parámetro String que indica la unidad a la que se quiere convertir, que puede ser “millas”, “pulgadas” o “pies”. El resultado es el valor obtenido en la conversión. Si el nombre de la unidad a convertir no es ninguna de las indicadas, el resultado es null.
- convertirPrimeraLetraAMayusculas: recibe como parámetro un texto y lo devuelve poniendo en mayúscula la primera letra de cada palabra. Sólo tiene en cuenta los signos de puntuación “,” y “.” que aparezcan en el texto.

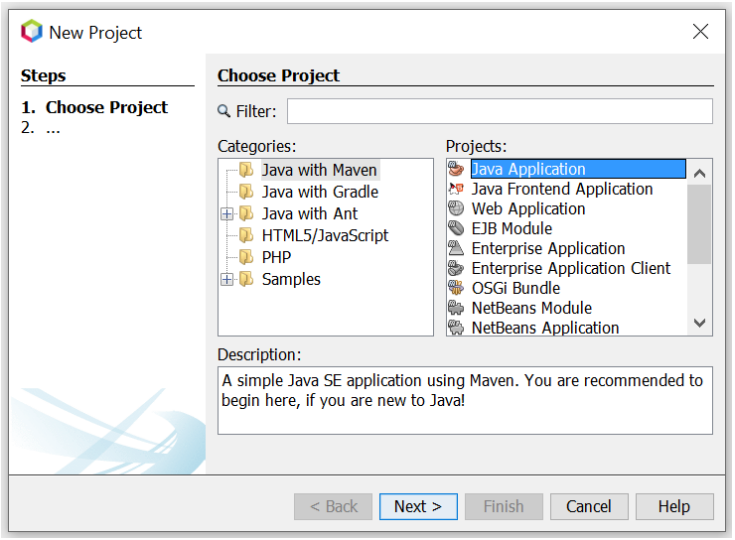
Programación de la clase a probar

La clase a probar es Conversor:

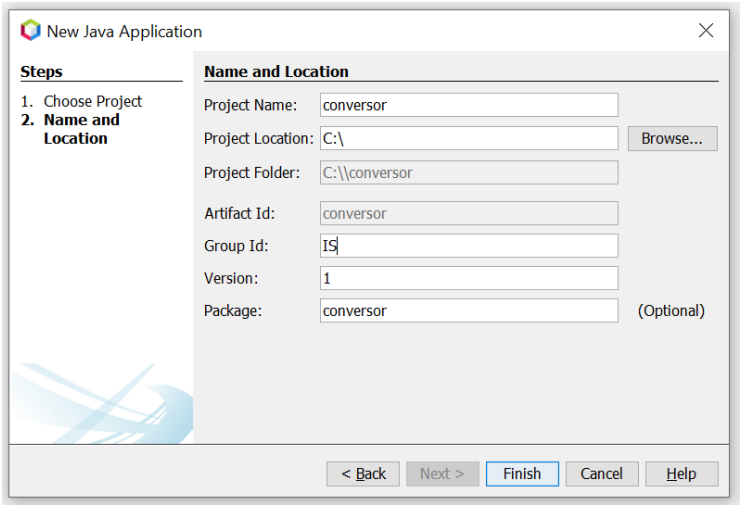


Los pasos a seguir para crear la clase en NetBeans son los siguientes.

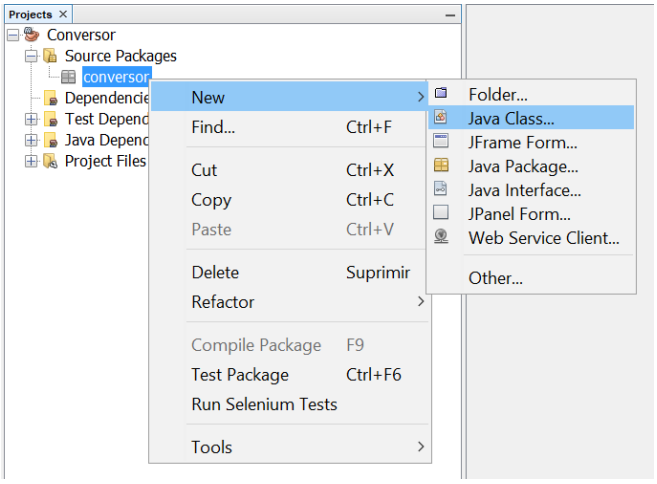
1. Crear un nuevo proyecto de tipo *Java Application*. Para esto se debe seleccionar en el menú: File > New Project > Java with Maven > Java Application

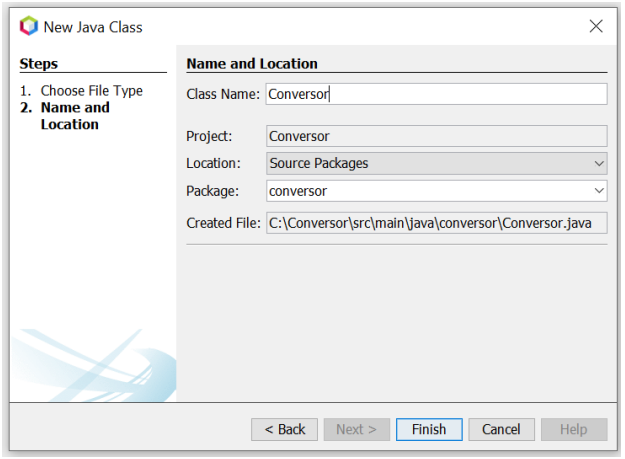


2. Llamar al proyecto “conversor” y ubicarlo en la carpeta que se quiera.



3. Añadir una clase “Conversor” al paquete “conversor”

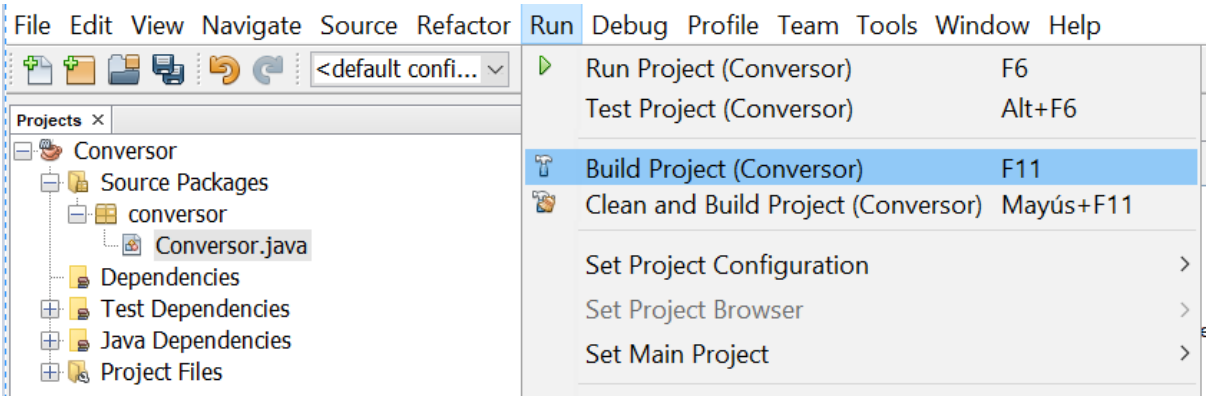




4. Y escribir el siguiente código:

```
package conversor;

public class Conversor {

    public double convertirMetros(double metros, String unidadDestino) {

        if (unidadDestino == "millas") {
            return (metros / 1609);
        } else if (unidadDestino == "pulgadas") {
            return (metros * 39.37);
        } else if (unidadDestino == "pies") {
            return (metros * 3.281);
        } else {
            return -1;
        }
    }

    public String convertirPrimeraLetraAMayusculas(String texto) {
        char[] caracteres = texto.toCharArray();
        for (int i = 0; i < texto.length() - 2; i++) {

            if (caracteres[i] == ' ' || caracteres[i] == '.' || caracteres[i] == ',') {
                caracteres[i + 1] = Character.toUpperCase(caracteres[i + 1]);
            }
        }
        return String.valueOf(caracteres);

    }

}
```

5. Compilar el proyecto pulsando F11 o Run > Build Project (Conversor)



Programacion de pruebas unitarias

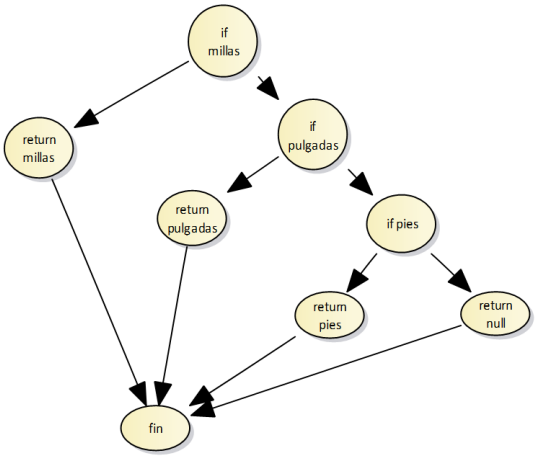
Diseño de los casos de prueba

Se trata de diseñar las pruebas unitarias de los dos métodos de la clase Conversor.
Lo primero que hay que hacer es calcular la **complejidad ciclomática** de cada método, que nos indicará el número mínimo de casos que hay que probar.
La ***complejidad ciclomática*** de McCabe es el número de caminos lógicos individuales contenidos en el código de un programa, un método o una función. Para calcular la complejidad ciclomática se puede representar el código en forma de grafo, cuyos nodos son las instrucciones, y los posibles caminos de ejecución sus aristas. Una vez representado de este modo, la complejidad ciclomática se calcula como:

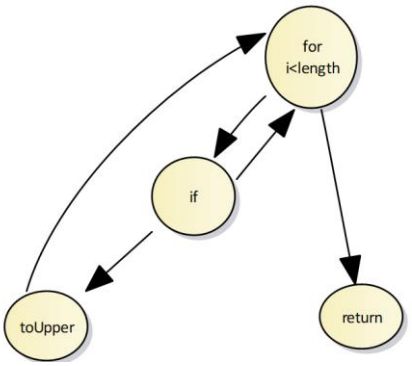
Complejidad ciclomática = Número de aristas – Numero de nodos + 2

Entonces para el ejemplo, se calcularía así:

- Método `convertirMetros`: 10 aristas – 8 nodos + 2 = 4. Hay que preparar 4 casos de prueba: uno para convertir a millas, otro a pulgadas, otro a pies y otro que devuelva un valor nulo.



- Metodo `convertirPrimeraLetraAMayusculas`: 5 aristas – 4 nodos + 2 = 3. Hay que preparar al menos 3 casos de prueba: uno con un texto vacío; uno con un texto sin ninguna palabra, pero con algún espacio, coma o punto; y otro con una o más palabras.



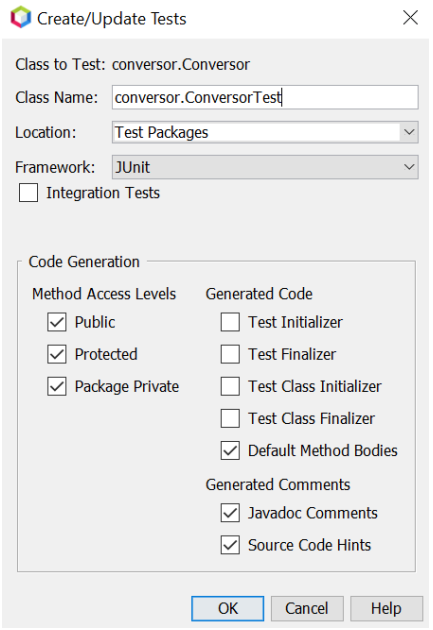
Programación de los casos de prueba

Crear una clase de prueba usando JUnit 5 para la clase `Conversor` que pruebe cada uno de sus métodos. Para esto hay que realizar los siguientes pasos:

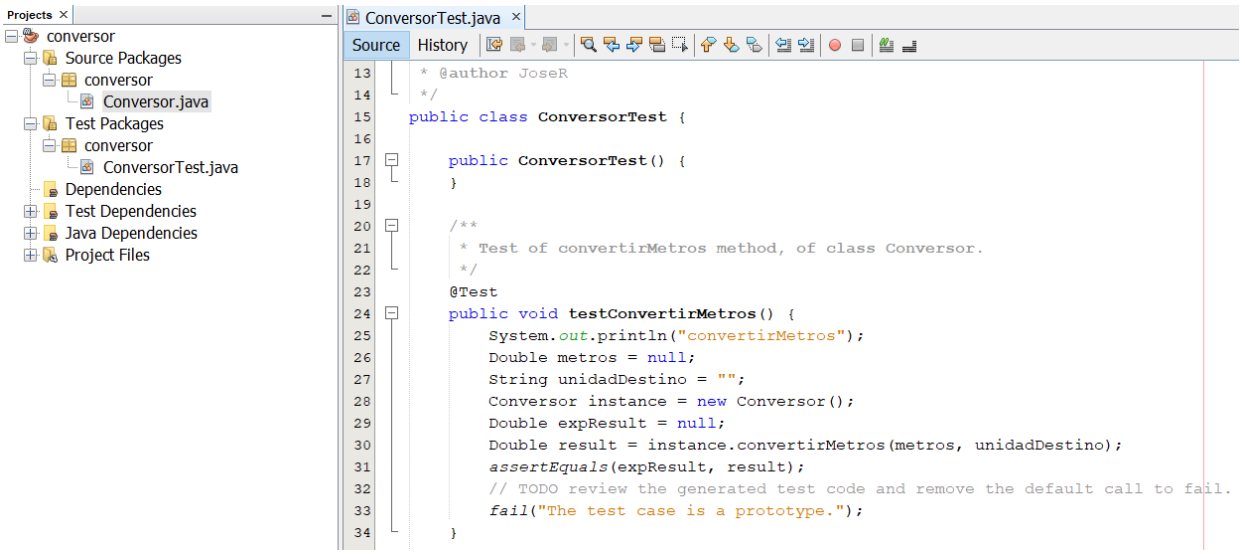
En la ventana *Projects* se selecciona la clase para la que se quiere construir el caso de prueba, que en este caso es la clase Conversor.java.

En la barra de menú seleccionar: Tools > Create/Update Tests.

Aparece la ventana emergente *Create/Update Tests* donde se define los parámetros para generar el esqueleto del caso de prueba. Verificar que no estén marcadas las opciones con el término Initializer ni Finalizer.



En la ventana del proyecto aparece una carpeta “Test Packages” con un archivo `ConversorTest.java`.



El archivo `ConversorTest.java`, incluye dos métodos de prueba, cada uno preparado para llevar a cabo la prueba de uno de los métodos de la clase `Conversor` que hay que probar, que se llaman igual pero con la expresión “test” delante del nombre del método.

Delante de cada método de prueba se indica la anotación `@Test`, para que JUnit reconozca que se trata de un método de prueba.

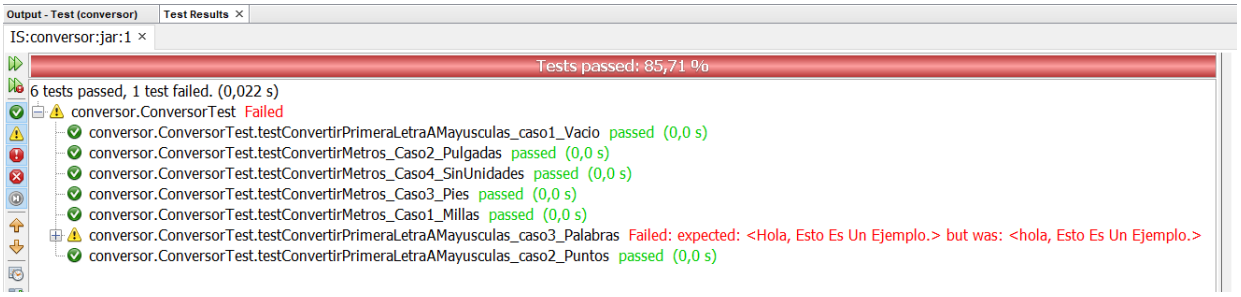
Hay que borrar el código generado por defecto, y preparar 4 casos de prueba para el método `convertirMetros` y 3 casos de pruebas para el método `convertirPrimeraLetraAMayusculas`, puesta esta era la complejidad ciclomática de ambos métodos.

El código final sería el siguiente:

```
package conversor;

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

public class ConversorTest {

    public ConversorTest() {
    }

    @Test
    public void testConvertirMetros_Caso1_Millas() {

        double metros = 10;
        Conversor unConversor = new Conversor();
        String unidadDestino;
        double resultadoEsperado, resultadoObtenido;

        System.out.println("convertirMetros - Caso de prueba con millas");
        unidadDestino = "millas";
        resultadoEsperado = metros / 1609;
        resultadoObtenido = unConversor.convertirMetros(metros, unidadDestino);
        assertEquals(resultadoEsperado, resultadoObtenido);
    }

    @Test
    public void testConvertirMetros_Caso2_Pulgadas() {

        double metros = 10;
        Conversor unConversor = new Conversor();
        String unidadDestino;
        double resultadoEsperado, resultadoObtenido;

        System.out.println("convertirMetros - Caso de prueba con pulgadas");
        unidadDestino = "pulgadas";
        resultadoEsperado = metros * 39.37;
        resultadoObtenido = unConversor.convertirMetros(metros, unidadDestino);
        assertEquals(resultadoEsperado, resultadoObtenido);
    }

    @Test
    public void testConvertirMetros_Caso3_Pies() {

        double metros = 10;
        Conversor unConversor = new Conversor();
        String unidadDestino;
        double resultadoEsperado, resultadoObtenido;

        System.out.println("convertirMetros - Caso de prueba con pies");
        unidadDestino = "pies";
        resultadoEsperado = metros * 3.281;
        resultadoObtenido = unConversor.convertirMetros(metros, unidadDestino);
        assertEquals(resultadoEsperado, resultadoObtenido);

    }

    @Test
    public void testConvertirMetros_Caso4_SinUnidades() {

        double metros = 10;
        Conversor unConversor = new Conversor();
        String unidadDestino;
        double resultadoEsperado, resultadoObtenido;

        System.out.println("convertirMetros - Caso de prueba con unidad desconocida");
        unidadDestino = "otra";
        resultadoEsperado = -1;
        resultadoObtenido = unConversor.convertirMetros(metros, unidadDestino);
        assertEquals(resultadoEsperado, resultadoObtenido);

    }


    @Test
    public void testConvertirPrimeraLetraAMayusculas_caso1_Vacio() {

        String texto;
        Conversor unConversor = new Conversor();
```

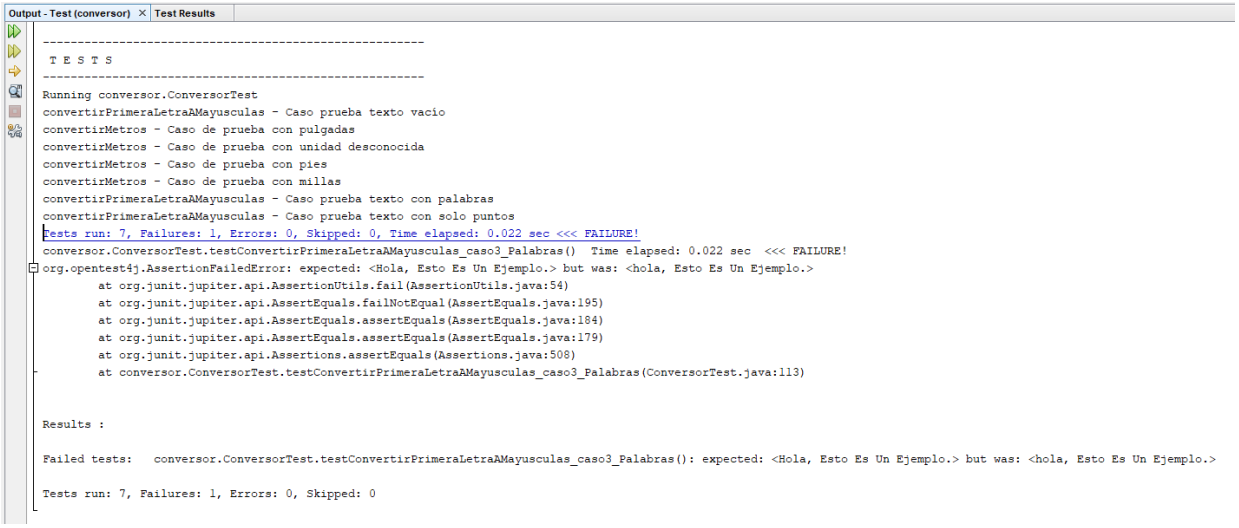
```
        String resultadoEsperado, resultadoObtenido;

        System.out.println("convertirPrimeraLetraAMayusculas - Caso prueba texto vacío");
        texto = "";
        resultadoEsperado = "";
        resultadoObtenido = unConversor.convertirPrimeraLetraAMayusculas(texto);
        assertEquals(resultadoEsperado, resultadoObtenido);
    }

    @Test
    public void testConvertirPrimeraLetraAMayusculas_caso2_Puntos() {

        String texto;
        Conversor unConversor = new Conversor();
        String resultadoEsperado, resultadoObtenido;

        System.out.println("convertirPrimeraLetraAMayusculas - Caso prueba texto con solo
puntos");
        texto = "...";
        resultadoEsperado = "...";
        resultadoObtenido = unConversor.convertirPrimeraLetraAMayusculas(texto);
        assertEquals(resultadoEsperado, resultadoObtenido);
    }

    @Test
    public void testConvertirPrimeraLetraAMayusculas_caso3_Palabras() {

        String texto;
        Conversor unConversor = new Conversor();
        String resultadoEsperado, resultadoObtenido;

        System.out.println("convertirPrimeraLetraAMayusculas - Caso prueba texto con
palabras");
        texto = "hola, esto es un ejemplo.";
        resultadoEsperado = "Hola, Esto Es Un Ejemplo.";
        resultadoObtenido = unConversor.convertirPrimeraLetraAMayusculas(texto);
        assertEquals(resultadoEsperado, resultadoObtenido);
    }

}
```

Para ejecutar los test hay que elegir en el menú principal Run > Test Project (Conversor). Y se muestra el resultado en la ventana Test Results:



Indica que de los 7 casos de prueba se han pasado 6 y uno ha fallado. Se trata del conversor de letras que no convierte bien la primera letra del texto. Gracias a JUnit nos damos cuenta de que se programó mal el método `convertirPrimeraLetraAMayusculas`, y habrá que revisar su código fuente para tratar ese caso.

Además de la ventana Test Result, también se pueden ver los mensajes que escribe cada método de prueba en la ventana Output:



Inicialización de valores antes de las pruebas

JUnit 5 permite ejecutar instrucciones antes de lanzar cada prueba utilizando las anotaciones siguientes:

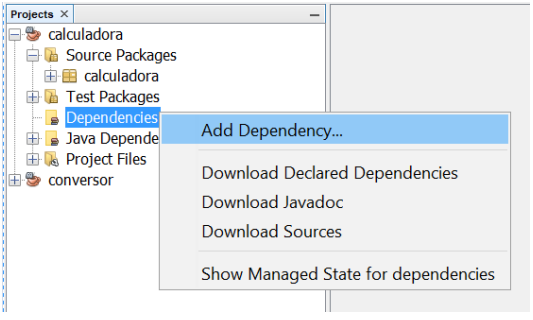
- @BeforeAll y a continuación un método, normalmente llamado setup(), que se ejecutará una vez antes que todos los métodos de prueba anotados como @Test.
- @BeforeEach y a continuación un método, normalmente llamado setup(), que se ejecutará justo antes de cada método anotado como @Test.
- @AfterAll y a continuación un método, normalmente llamado finish(), que se ejecutará una vez después de terminar todos los métodos de prueba anotados como @Test.
- @AfterEach y a continuación un método, normalmente llamado finish(), que se ejecutará justo después de cada método anotado como @Test.

Se recomienda consultar el artículo "Intro to Unit Testing in Java With JUnit5 Library", y tratar de utilizarlo en el ejemplo anterior.

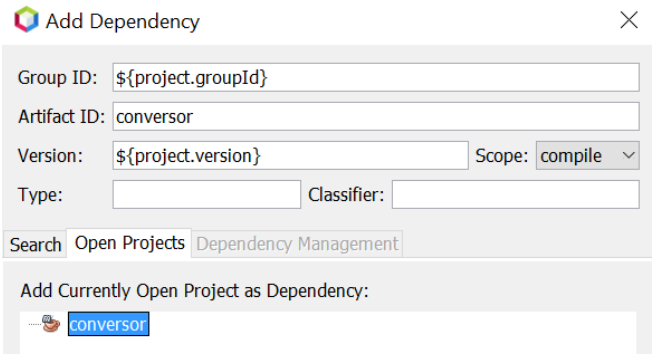
Programación de pruebas de integración

Las pruebas de integración tratan de probar la integración de varios componentes. En el siguiente ejemplo, supongamos que se crea un nuevo proyecto Java para programar una clase Calculadora que utiliza la clase Conversor ya creada en el apartado anterior y disponible en la librería conversor.jar.

El proyecto calculadora se crearía como el proyecto del conversor. A continuación, para poder utilizar la librería conversor se selecciona Dependencies > Add Dependency



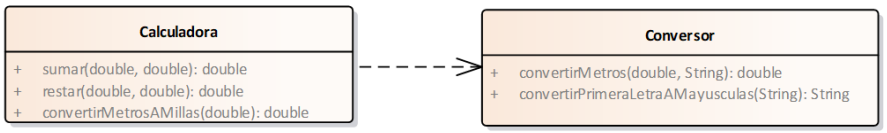
Y se elige el proyecto conversor:



Y el código de la clase Calculadora podría ser:

```
package calculadora;

import conversor.Conversor;


public class Calculadora {

    public double sumar(double a, double b) {
        return a + b;
    }

    public double restar(double a, double b) {
        return a - b;
    }

    public double convertirMetrosAMillas(double m) {
        Conversor c = new Conversor();
        return c.convertirMetros(m, "millas");
    }

}
```

El diagrama de clases conjunto es el siguiente, en el que se indica que la clase Calculadora depende de la clase Conversor:

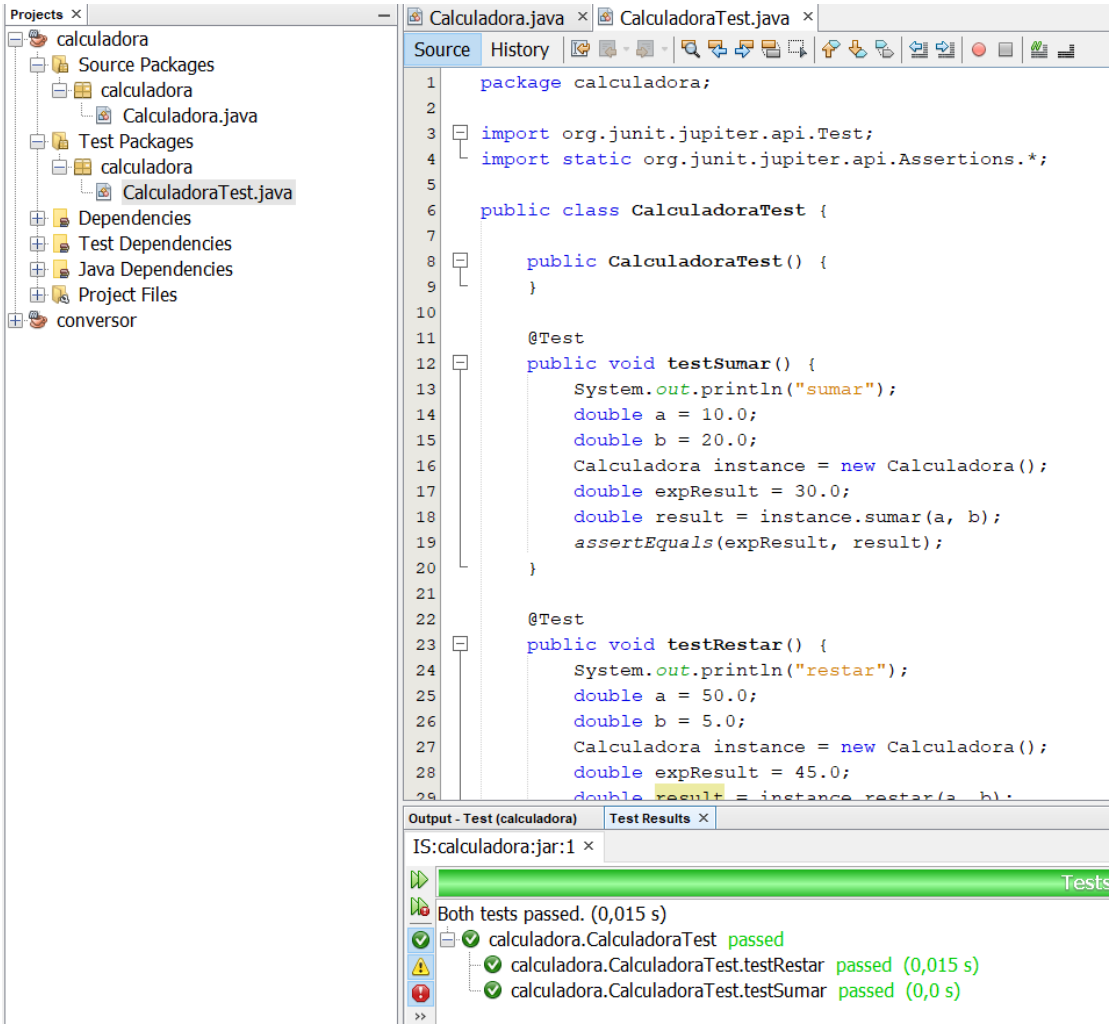


Pruebas unitarias

Se podrían programas pruebas unitarias para los métodos de suma y resta de la clase Calculadora:

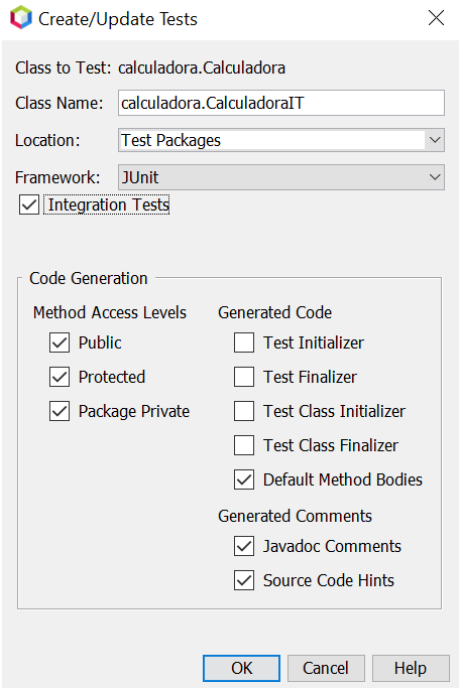
```
package calculadora;

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

public class CalculadoraTest {

    public CalculadoraTest() {
    }

    @Test
    public void testSumar() {
        System.out.println("sumar");
        double a = 10.0;
        double b = 20.0;
        Calculadora instance = new Calculadora();
        double expResult = 30.0;
        double result = instance.sumar(a, b);
        assertEquals(expResult, result);
```

```
    }

    @Test
    public void testRestar() {
        System.out.println("restar");
        double a = 50.0;
        double b = 5.0;
        Calculadora instance = new Calculadora();
        double expResult = 45.0;
        double result = instance.restar(a, b);
        assertEquals(expResult, result);
    }
}
```



Pruebas de integración

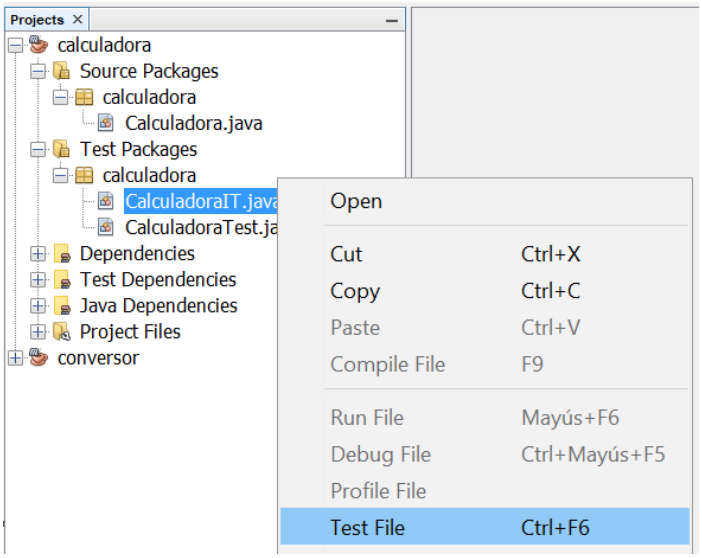
Para separar las pruebas de integración de las unitarias, se puede volver a generar una clase de prueba para Calculadora, pero esta vez eligiendo “Integration Test” al crear el test.



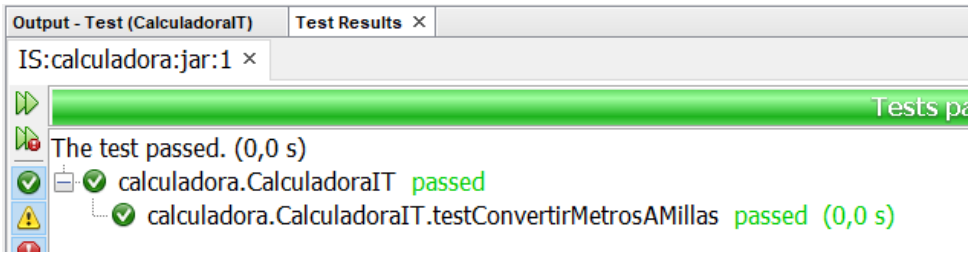
El código de la prueba podría ser:

```
package calculadora;

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

public class CalculadoraIT {

    public CalculadoraIT() {
    }

    @Test
    public void testConvertirMetrosAMillas() {
        System.out.println("convertirMetrosAMillas");
        double m = 1000.0;
        Calculadora instance = new Calculadora();
        double expResult = 1000.0/1609.0;
        double result = instance.convertirMetrosAMillas(m);
        assertEquals(expResult, result);
    }

}
```

Las pruebas de integración no se ejecutan en NetBeans seleccionando Run > Test Project, sino que hay que seleccionar la clase de prueba CalculadoraIT, pulsar el botón derecho del ratón y elegir Test File.



Y la prueba se pasa sin problemas:



EJERCICIO PROPUESTO – Pruebas unitarias

1. Corregir el código necesario en el método `convertirPrimeraLetraAMayusculas` de la clase `Conversor` para que se pase la prueba que ha fallado.
2. Realizar modificaciones en el método `convertirMetros` de la clase `Conversor` para que falle la prueba de integración, y después corregirlo.
3. Tratar de utilizar algunas de las anotaciones @BeforeAll, @BeforeEach, @AfterAll, o @AfterEach en la clase de prueba `ConversorTest`. Se recomienda consultar el artículo "Intro to Unit Testing in Java With JUnit5 Library".