

## UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

### Facultad de Ciencias Químicas e Ingeniería

Programas de Ingeniero en Computación e Ingeniero en Software y Tecnologías Emergentes

#### INFORMACIÓN DE LA MATERIA

Nombre de la materia y clave: Herramientas de Desarrollo de Software (40017).

Grupo y periodo: 341 (2022-2)

Profesor: Manuel Castañón Puga.

#### INFORMACIÓN DE LA ACTIVIDAD

Nombre de la actividad: Actividad de taller 4.1.2: Exploración de Maven/Gradle para para validación y pruebas.

Lugar y fecha: A 9 de noviembre de 2022 en el Edificio 6E, Salón 204.

Carácter de la actividad: Individual

Participante(es): Diego Andrés González Beltrán

#### REPORTE DE ACTIVIDADES

Objetivo de la actividad:

En esta actividad se tiene como objetivo la exploración de la herramienta JUnit para pruebas unitarias. En mi caso, utilicé el IDE de IntelliJ y descargué la extensión de JUnit Generator V2.0 para dar clic a Generate... sobre la clase con métodos que quiero probar. Y dar clic a la versión que quiera de JUnit, en mi caso use JUnit 4 (Figura 1).

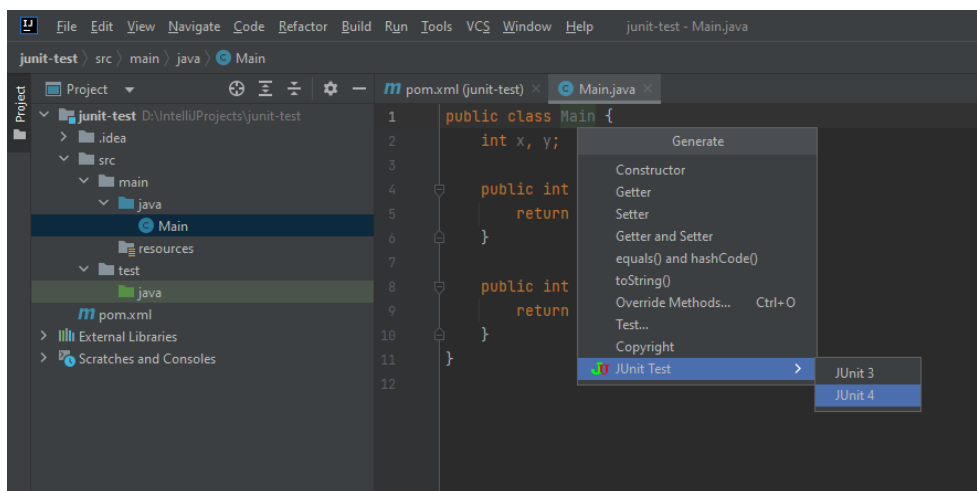


Figura 1. Generando la clase MathUtilsTest con JUnit 4

En MathUtils existen dos métodos a probar, sum y diff, sum suma dos valores y diff resta dos valores. Ahora en la clase MathUtilsTest simplemente agregamos @Test a los métodos que quisiéramos probar, y utilizamos los métodos de Assert respectivos que queramos.

assertEquals() toma el valor que resultará de la operación que realiza el método a probar y lo compara con el valor esperado, este valor esperado lo podemos asignar desde un inicio para

utilizarla precisamente en assertEquals. En el caso del test con valor esperado de 3, resultó correcto porque la suma que se realizó fue de  $1 + 2 = 3$ . Y en el método testDiff(), se esperaba un 0 de la operación  $1 - 2$ , esto es incorrecto y por eso se muestra en consola la excepción.

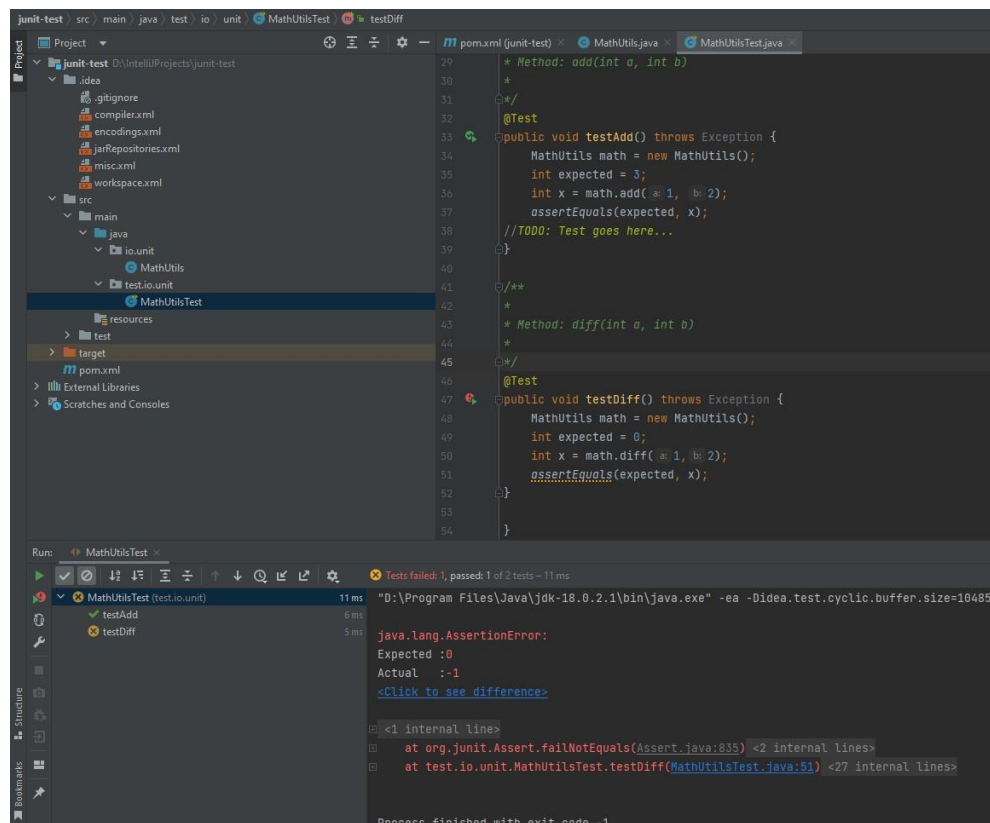


Figura 2. Métodos de testing para los métodos de MathUtil.


Esto ha sido posible por el framework de Maven que nos ofrece el archivo pom.xml, este archivo es donde se asignan las dependencias para frameworks de validación y de pruebas como JUnit. Maven es 100% para Java mientras que Gradle es posible con distintos lenguajes y no requiere de un archivo pom.xml como lo hace Maven.

### RESUMEN/REFLEXIÓN/CONCLUSIÓN

En esta actividad se utilizaron pruebas unitarias con JUnit para probar y comprobar métodos en las clases con la ayuda de Maven.

Aprendí como funciona la herramienta de Maven ya que ofrece los plugins y el archivo pom.xml necesarias para crear dependencias de los frameworks de testing como JUnit.

Concluyo que Maven es una gran herramienta pero a diferencia de Gradle, este solo es compatible con Java y requiere de un archivo pom.xml para agregar dependencias. Afortunadamente por utilizar un IDE como IntelliJ crear proyectos con build systems como Maven es mucho más fácil y también extensiones de frameworks de testing como JUnit.

<p>Doy fe de que toda la información dada es completa y correcta.</p>	<p>Diego Andres Gonzalez Beltran</p> 
---	--