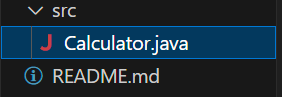
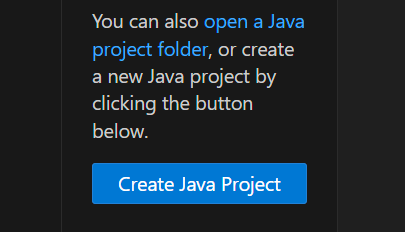
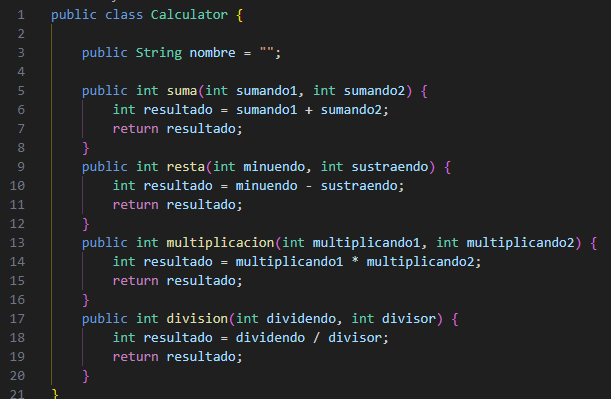
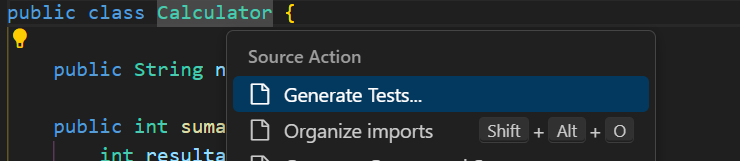
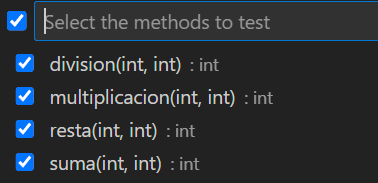
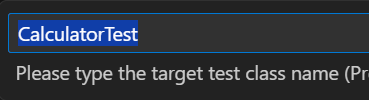
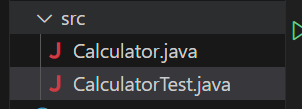
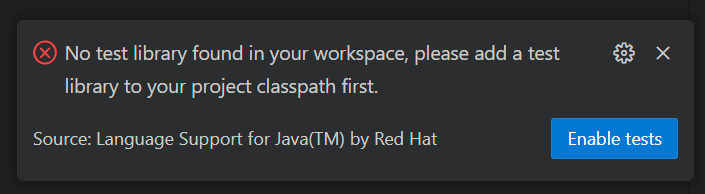
Tarea: Depuración y pruebas

# Ejercicio 1: Calculadora Simple

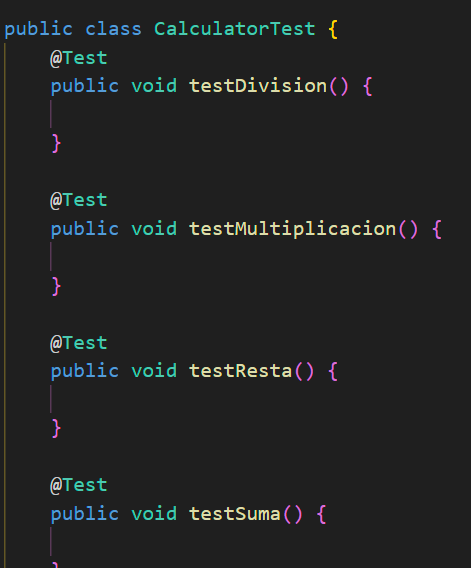
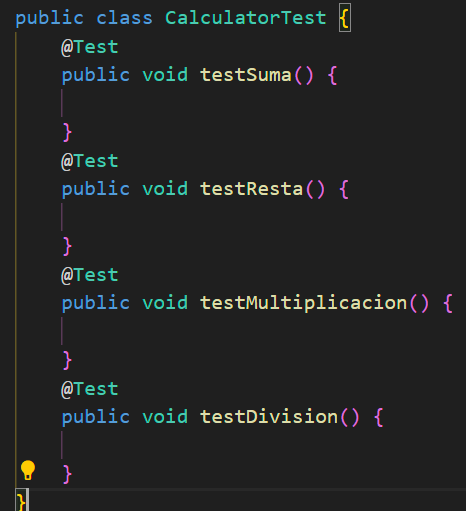
Lo primero que se debe hacer es crear un proyecto de java nuevo, en él se crea un archivo .java que será donde se escribirá el código del programa de la calculadora, con todos los métodos requeridos, así como nombre de atributos diferentes para los datos de los diferentes métodos para poder diferenciar los tests bien y de manera clara y concisa posteriormente.



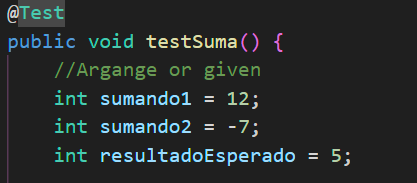
Una vez el programa de la calculadora con todos sus métodos esté completado se debe habilitar y generar los test, lo que dará lugar a un nuevo archivo .java, que se podrá nombrar, en el que se almacenarán los tests para los diferentes métodos previamente creados. 



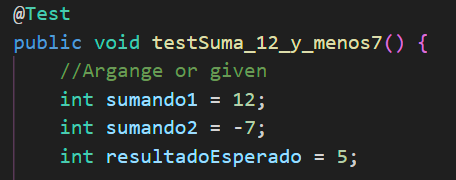
Dentro de este archivo deberá antes de nada ordenarse los diferentes tests ya automáticamente creados por Visual Studio según su orden real en el archivo de la calculadora, puesto que por defecto viene con otro orden y por tanto es menos intuitivo.

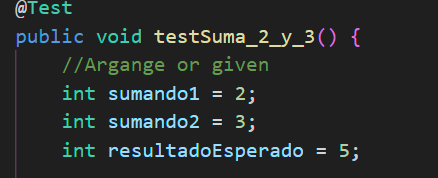


Una vez hecho eso, se procederá a elaborar el código para cada uno de los tests, el cual en este ejercicio concreto será muy similar para todos los métodos, ya que se trata simplemente de operaciones básicas. Para ello lo primero será dividir el proceso del test en 3 etapas: Arrange/given, la sección donde se establecen los datos que se usarán para realizar el test así como resultantes esperados en base a ellos o datos similares, además de inicializar la calculadora.



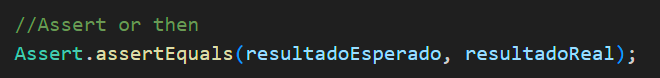
Puesto que con este tipo de test solo se está comprobando una suma particular entre 2 valores establecidos, y no el funcionamiento de la calculadora en base a otros medios, sería preciso añadir en el nombre además una especificación del cálculo que se está haciendo, puesto que en caso de hacerse más tests de suma con otros valores diferentes, lo cual sería útil de hacer para comprobar que el método funciona correctamente y no ha sido coincidencia, no habría forma de diferenciar ambos tests.



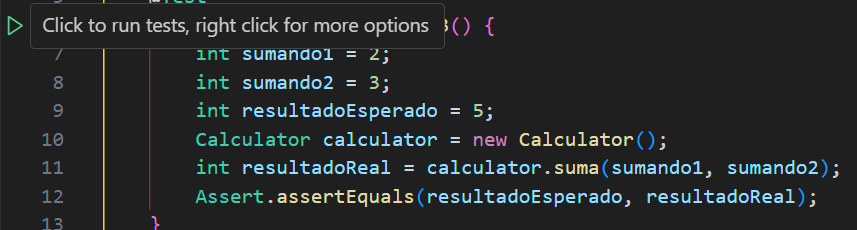
**(En base a esto deberían hacerse al menos 2 tests diferentes para cada método)**

2º Etapa: Act/when, la parte del código donde se realiza el cálculo/comprobación del código original, debido a haber inicializado la calculadora previamente, ahora es posible utilizar el método suma del código original, usando como datos los números establecidos arriba, dando lugar a el resultado que el código original da como respuesta, obviando el resultado esperado.



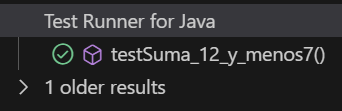
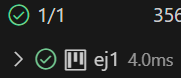
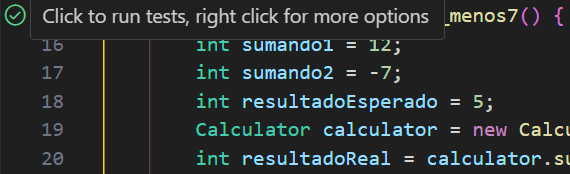
3º Etapa: Assert/then, donde se realiza la comprobación entre el resultado esperado, el que debería dar el programa si estuviese bien estructurado, y el resultado real que este ofrece. 

Para está última etapa se utiliza un assert, es decir, métodos que trae la librería de JUnit que permiten comprobar si el resultado es correcto, debe ser importado. 

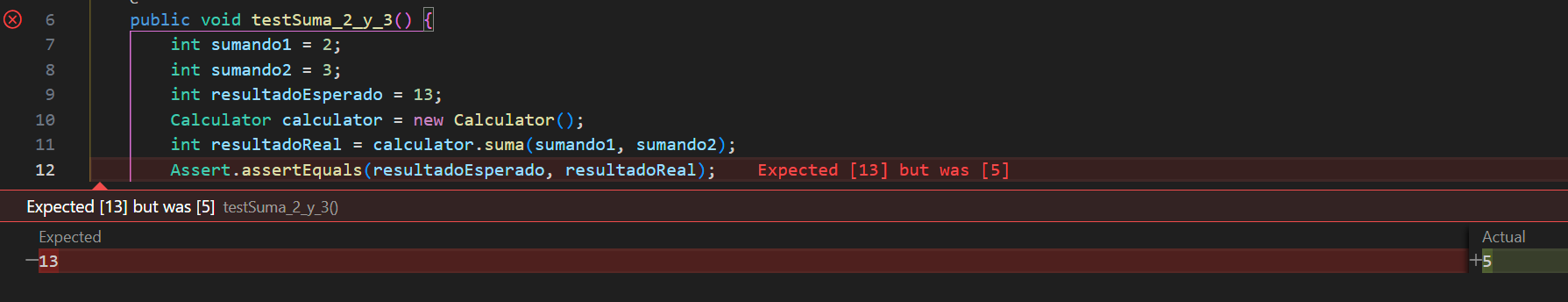
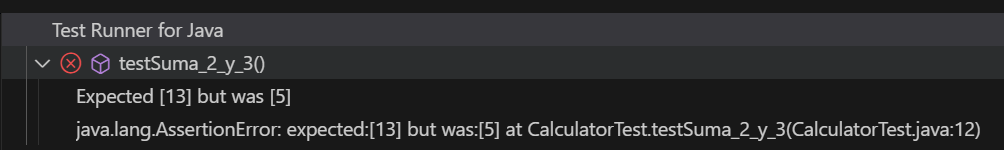
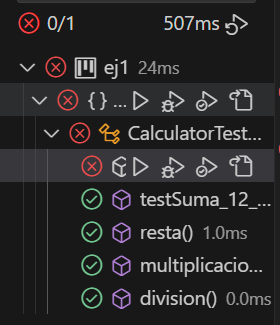
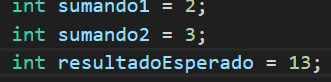
Una vez realizada la comprobación del código con el test, pudiendo hacerlo individualmente test a test, o globalmente todos a la vez, Visual Studio dará el resultado, en caso correcto, no ocurrirá nada y simplemente aparecerá un tick verde al lado del test indicando que es correcto, en caso de fallo, saldrá un aviso que mostrará que la comprobación a fallado, señalando cual es el resultado esperado en comparación con el que el programa a dado. 

**Test Global:**

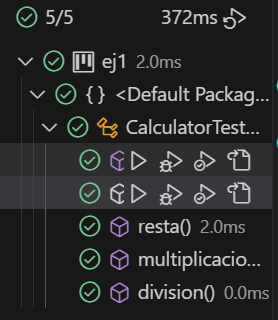
**Test correcto:**



**Test fallido:**



Si tras todo esto finalmente se comprueba mediante cada uno de los test que el código funciona en su completitud bien, habrá finalizado la fase de pruebas y el código estará listo.



# Ejercicio 2: Testing y Debugging