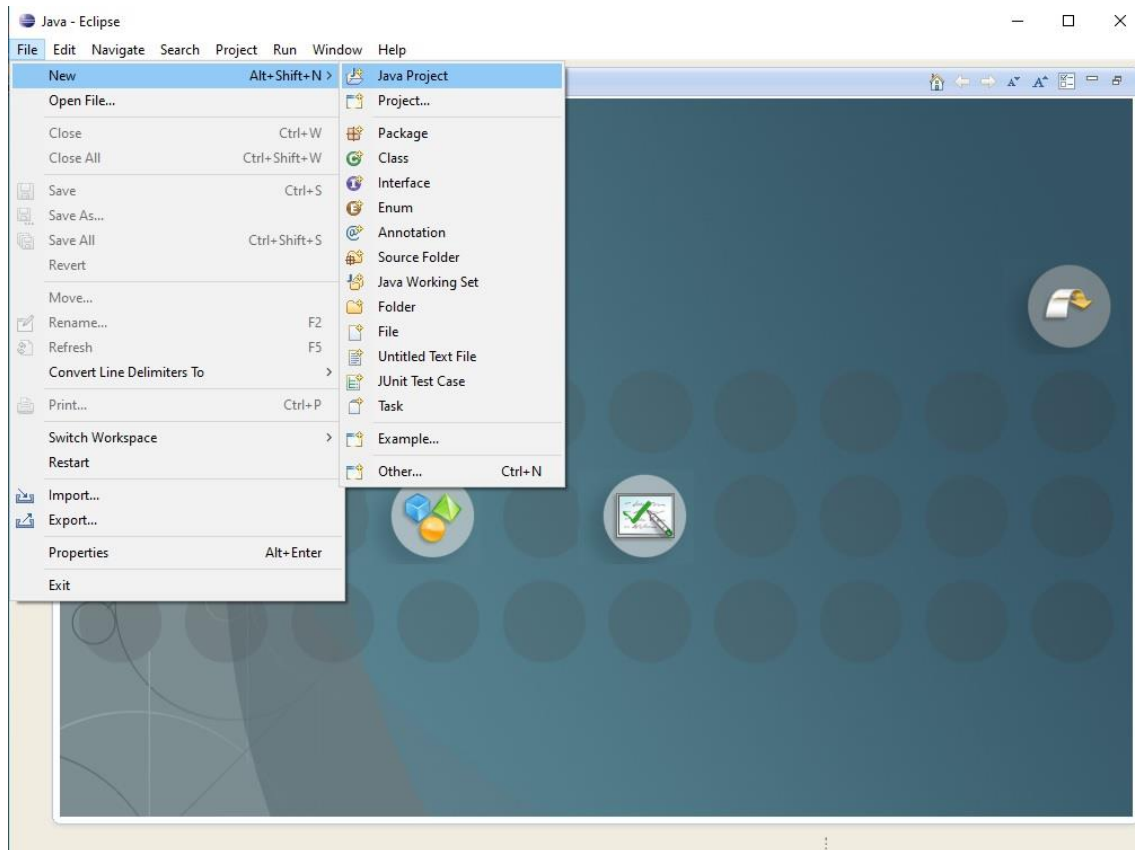
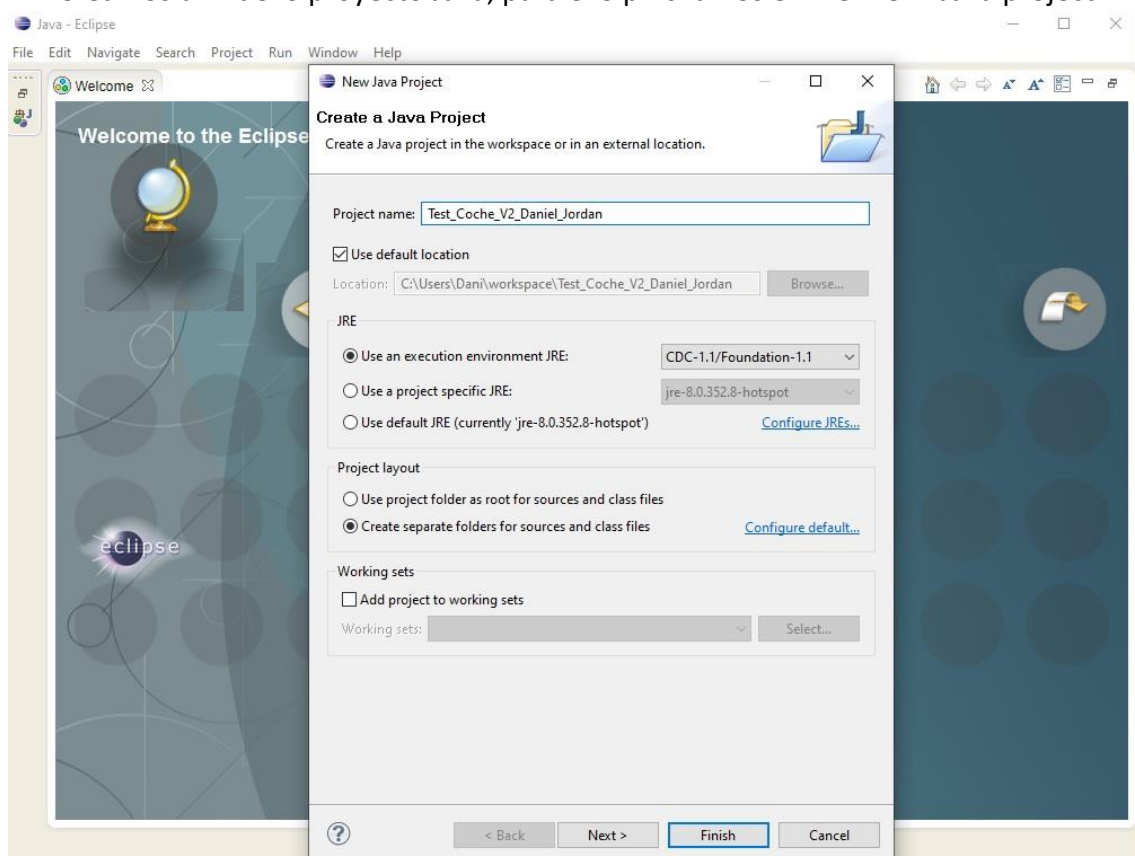


MEMORIA ENTORNOS DESARROLLO

TDD V2.0

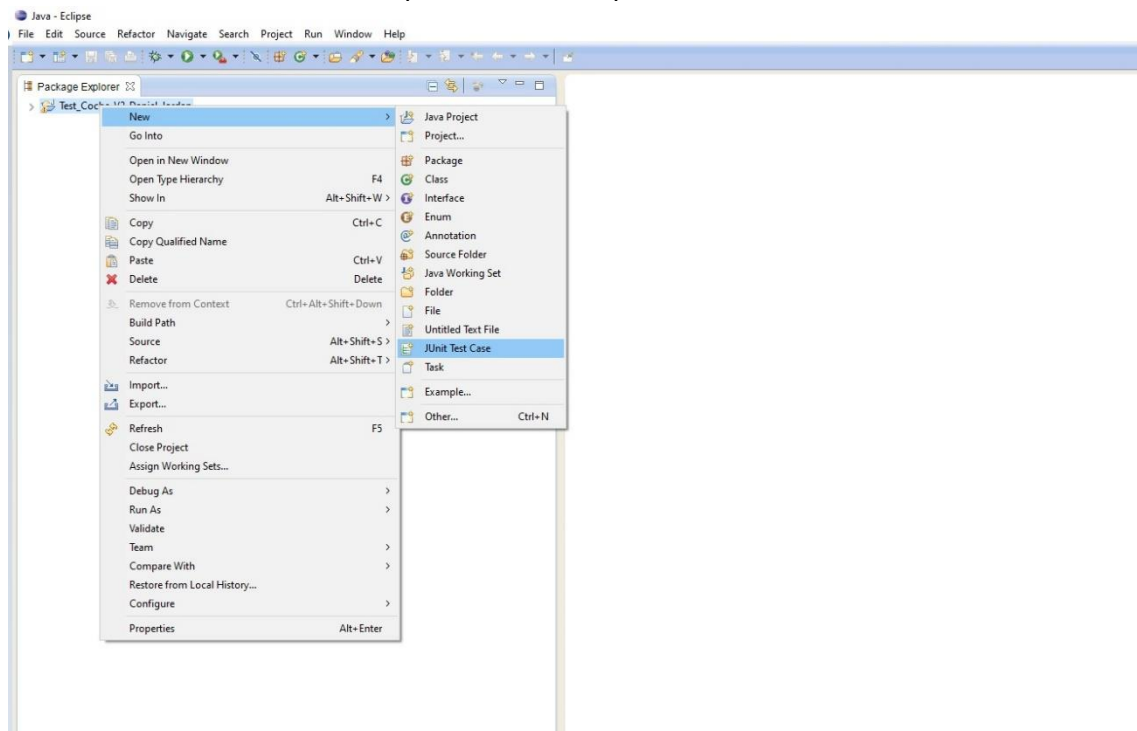


Creamos un nuevo proyecto Java, para ello pinchamos en file>new>Java project

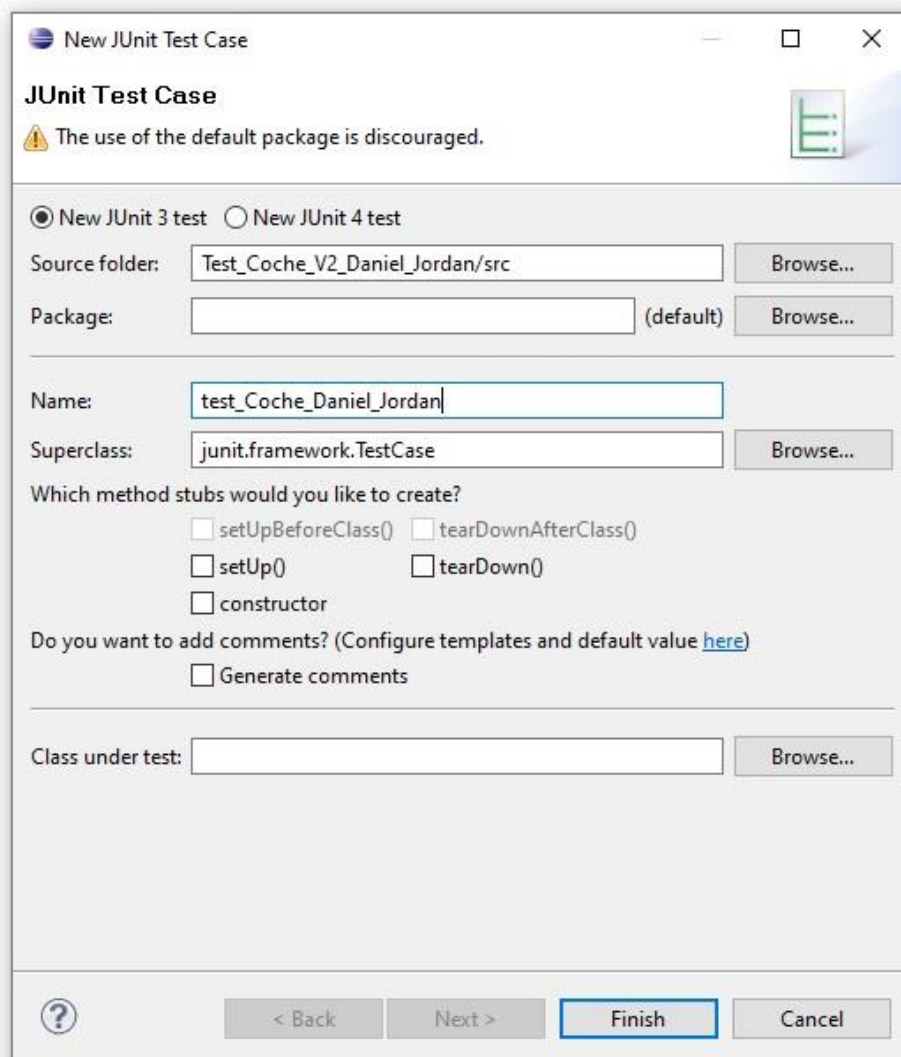


Nombramos el proyecto en mi caso esta es una captura previa de creación de proyecto con lo que no cuadrará el nombre de creación con el que veremos mas adelante ya

que me encontré con problemas en el entorno de desarrollo y tuve que volver a inicializar el proyecto esta vez cambiando la opción JRE y indicándole otro 'environment' exactamente el JSE-1.7 ya que este de la imagen al parecer me daba muchos problemas al importar las librerías.

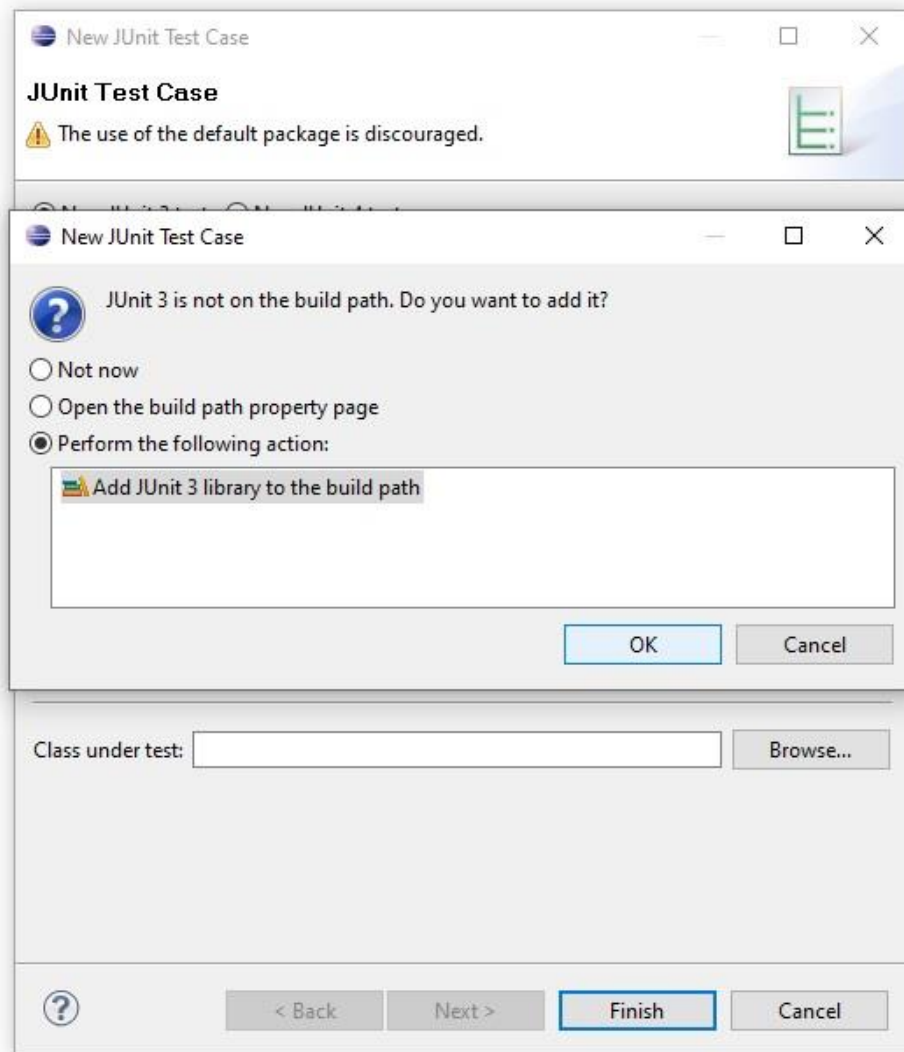


damos botón derecho en el proyecto y seleccionamos nuevo>JUnit Case. En la siguiente ventana vamos a crear el nombre del test que vamos a realizar.



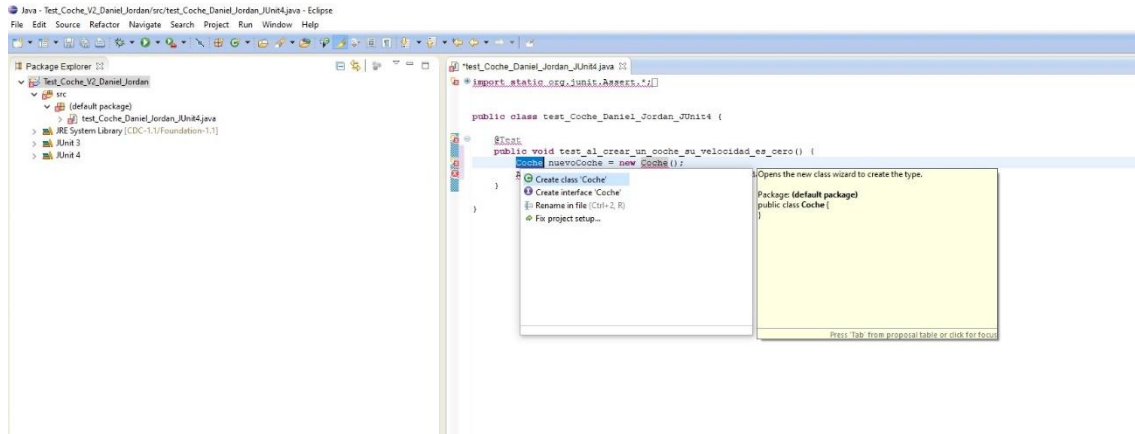
The screenshot shows the 'New JUnit Test Case' dialog box in Eclipse. The title bar reads 'New JUnit Test Case'. Below the title bar, there is a warning icon and the text 'The use of the default package is discouraged.' followed by an Eclipse logo. The dialog is divided into several sections. The first section has two radio buttons: 'New JUnit 3 test' (selected) and 'New JUnit 4 test'. Below this, there are two text fields: 'Source folder:' with the value 'Test_Coche_V2_Daniel_Jordan/src' and a 'Browse...' button, and 'Package:' with the value '(default)' and a 'Browse...' button. The next section has a 'Name:' text field with the value 'test_Coche_Daniel_Jordan'. Below that is a 'Superclass:' text field with the value 'junit.framework.TestCase' and a 'Browse...' button. The following section is titled 'Which method stubs would you like to create?' and contains five checkboxes: 'setUpBeforeClass()', 'tearDownAfterClass()', 'setUp()', 'tearDown()', and 'constructor'. The next section is titled 'Do you want to add comments? (Configure templates and default value [here](#))' and contains a checkbox labeled 'Generate comments'. The final section has a 'Class under test:' text field and a 'Browse...' button. At the bottom of the dialog, there is a row of buttons: a help button (question mark icon), '< Back', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

Aquí seleccionamos el nombre del test y pulsamos a finish.

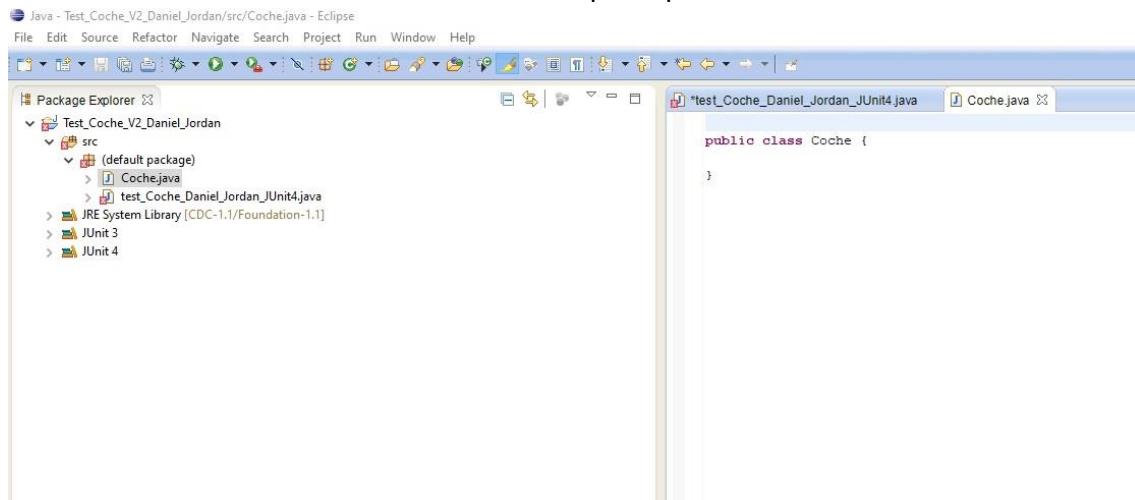


En la siguiente ventana seleccionamos la librería que queremos añadir para el test Junit. En mi caso como bien estoy comentando al encontrarme con problemas la primera vez lo intenté con Junit 3 y en la prueba de los test satisfactorios usé Junit4.

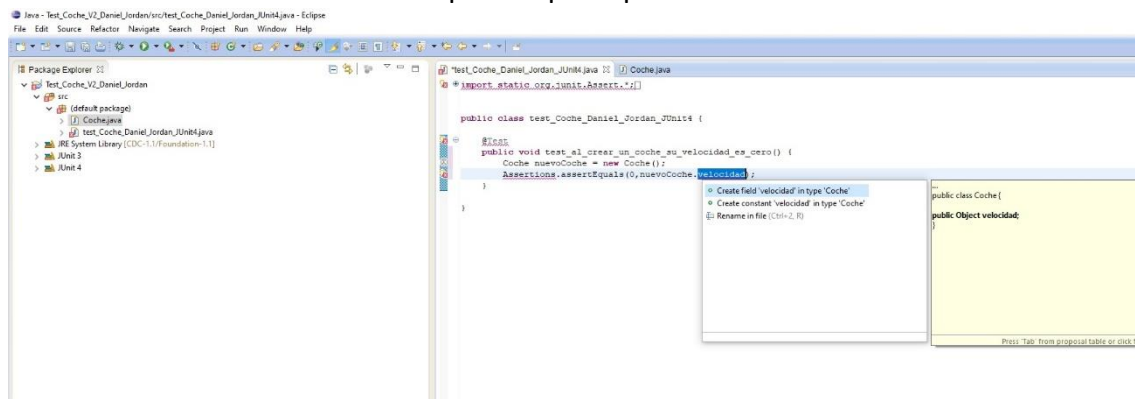
Una vez que tenemos



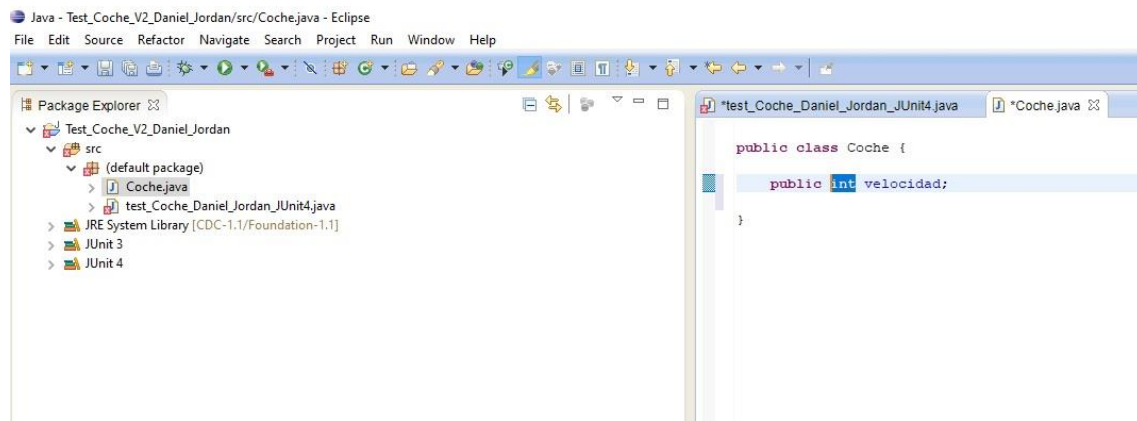
Modificamos el nombre del test por “test al crear un cohche su velocidad es cero” y creamos un objeto coche dentro del método. Así como el Assert que determina que el test inicial la velocidad que esperamos es “0”.



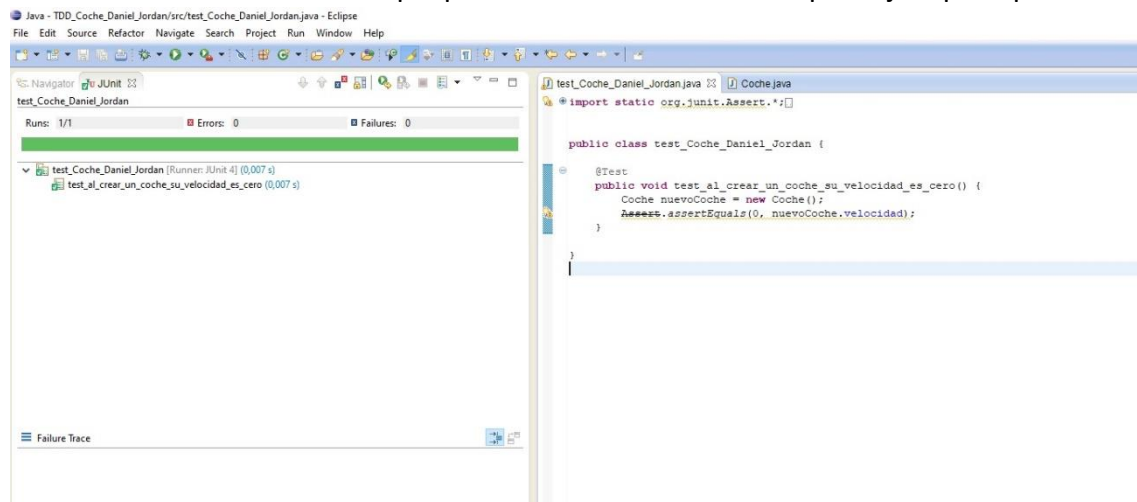
Como la clase coche no existe le damos a la crucecita roja que aparece a la izquierda de la creación del objeto coche y ahí nos invita a crear la clase coche y clicamos con botón izquierdo para que nos la cree.



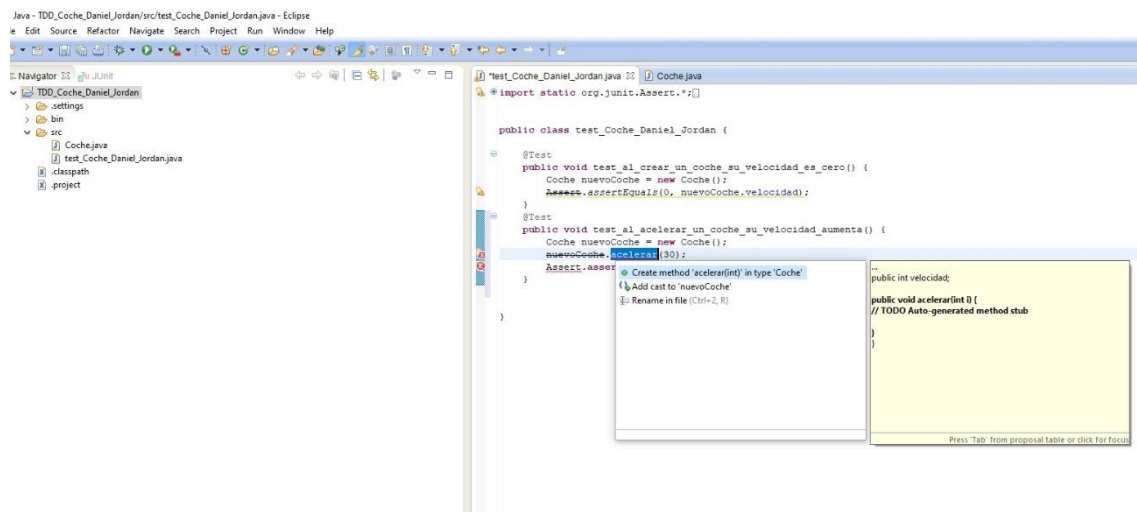
También debemos de crear el atributo de tipo entero velocidad, para ello pinchamos en la crucecita que hay en la izquierda del Assert y nos invitará a crear el atributo velocidad en la clase coche.



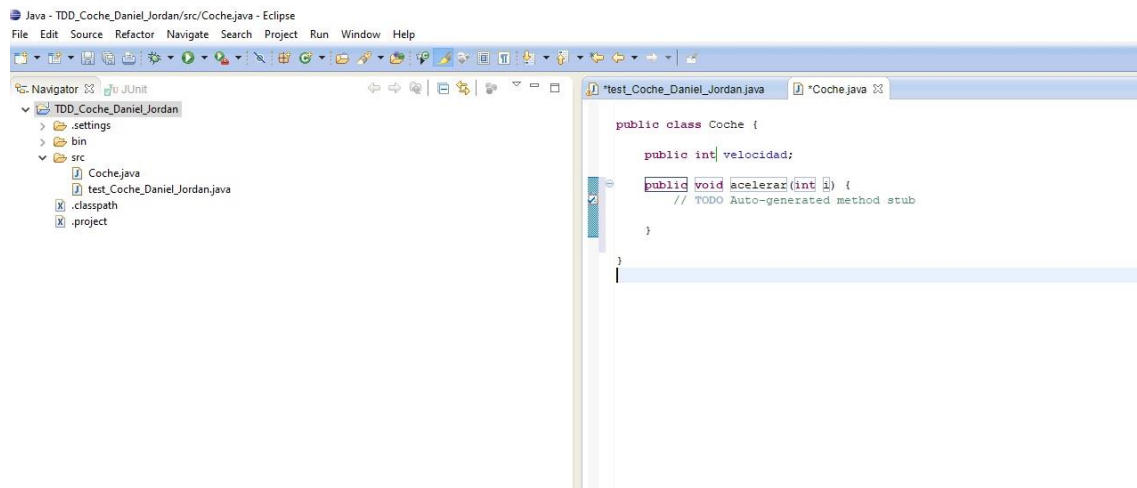
Modificamos el atributo que por defecto nos lo crea de tipo Object por tipo int



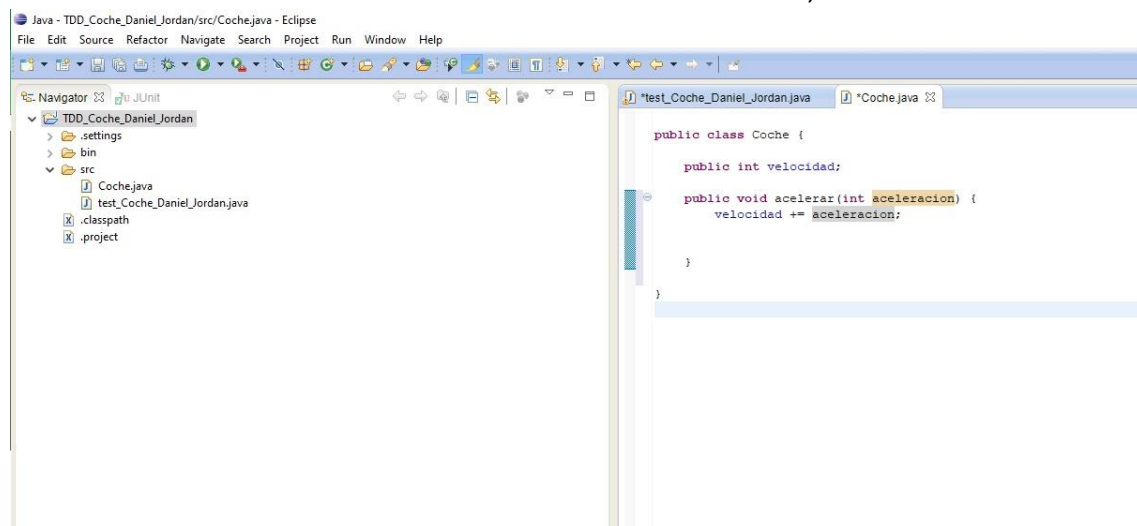
Y probamos el test con el botón play verde del menú superior y comprobamos que el test ha sido exitoso.



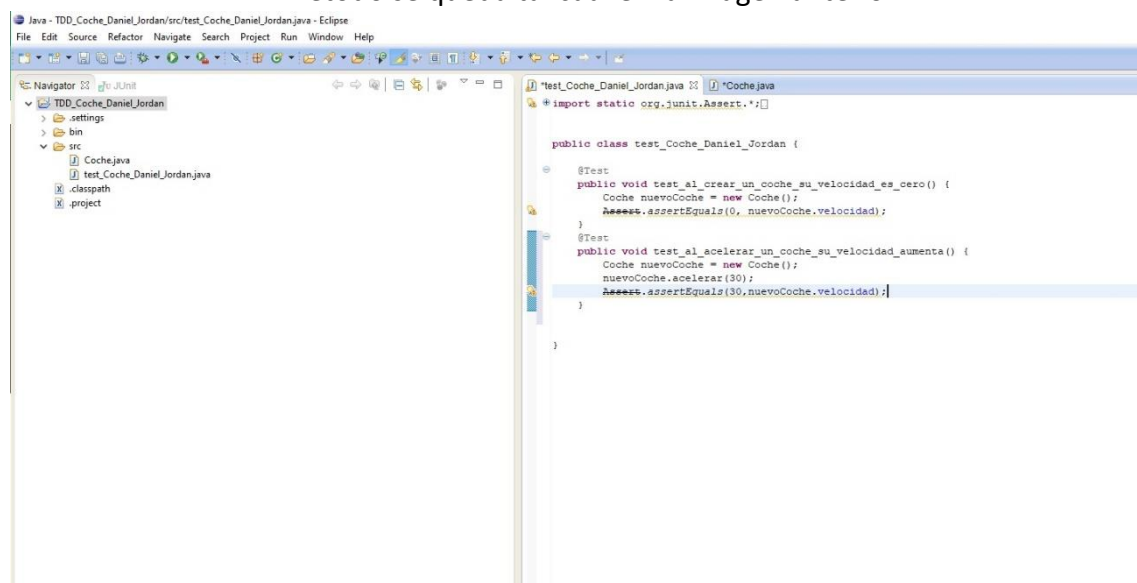
Ahora vamos a copiar el anterior método y lo renombramos por “al acelerar un coche su velocidad aumenta” y dentro del método le indicamos nuevoCoche.acelerar(30) para darle una aceleración de 30 al vehículo. Como el método acelerar no esta creado en la clase coche si ponemos el cursor encima de acelerar y en la crucecita roja que se nos marca a mano izquierda del método y la clicamos, veremos que el entorno nos invita a crear el método acelerar en la clase coche.



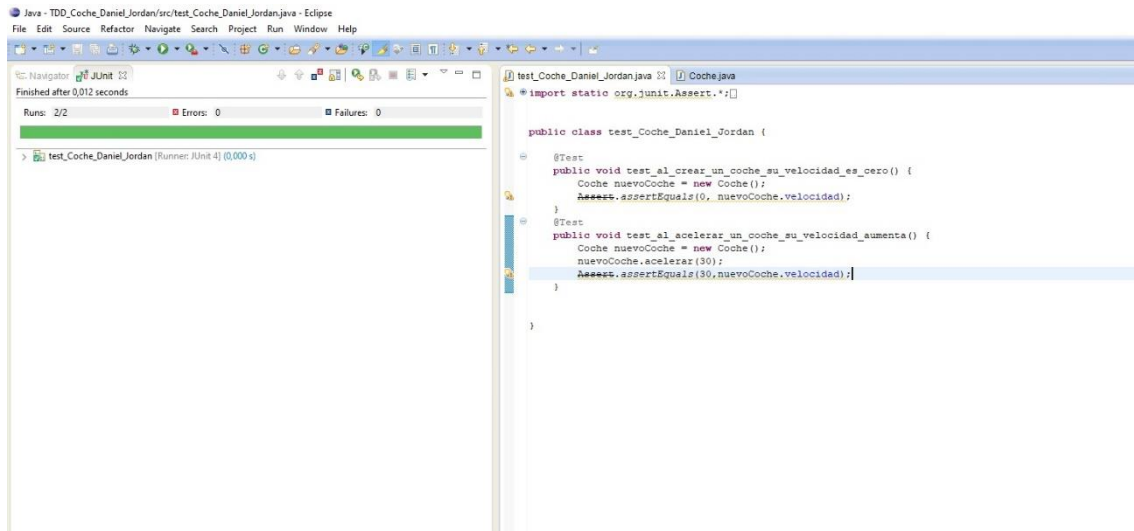
Una vez creado la variable que se le pasa nos la pone de tipo int lo cual en nuestro caso es correcto, y la “i” la modificamos por “aceleración”. Ahora dentro del método escribimos `velocidad += aceleración;`



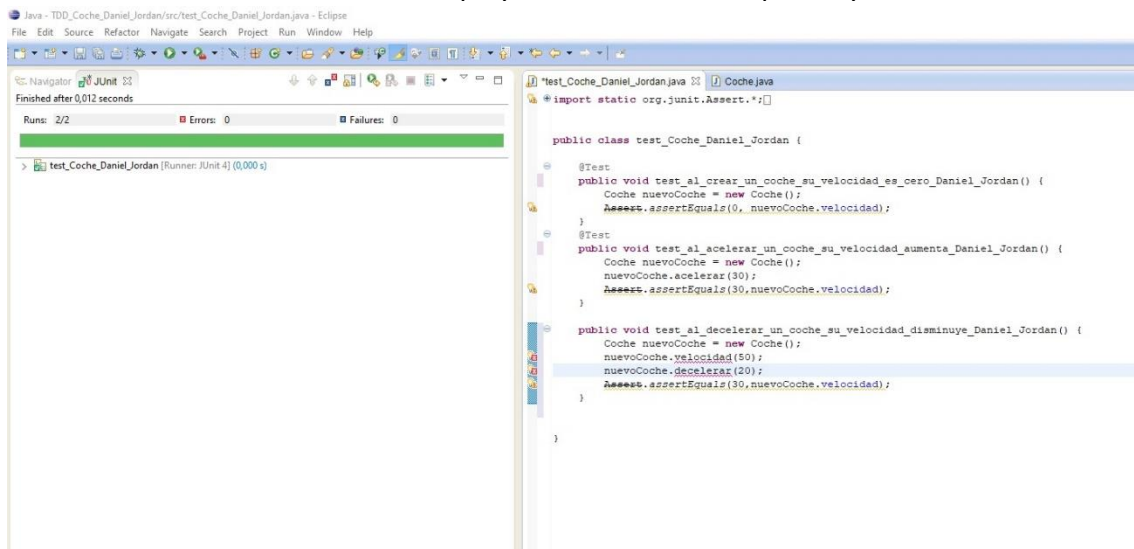
El método se queda tal cual en la imagen anterior.



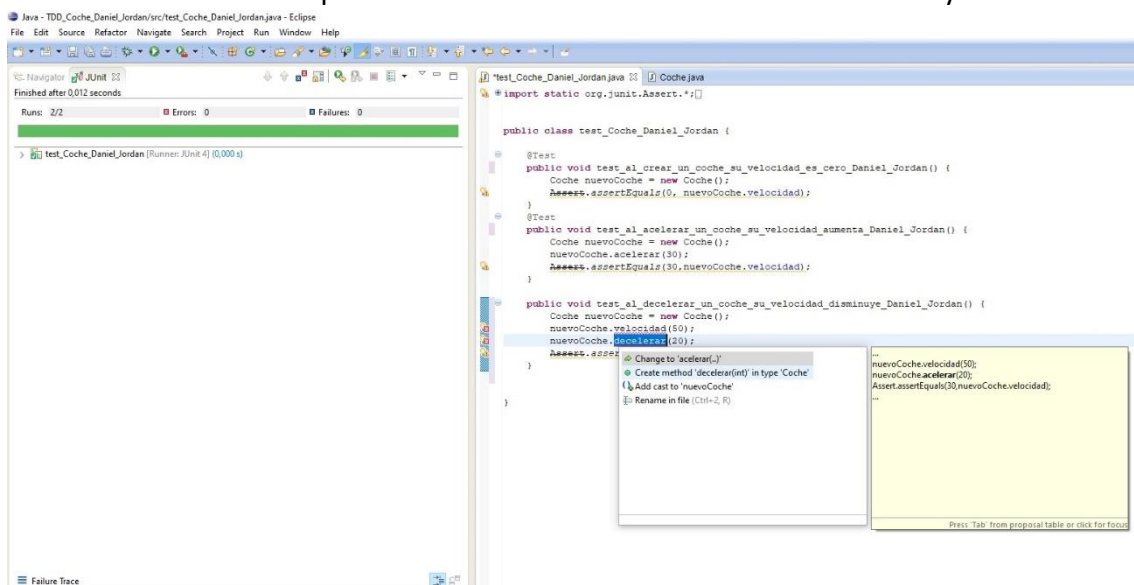
Al assert del método debemos indicarle que esperamos una velocidad de 30 despues de pasarle esa velocidad.



Probamos el test con el botón play verde del menú superior y es todo un éxito.

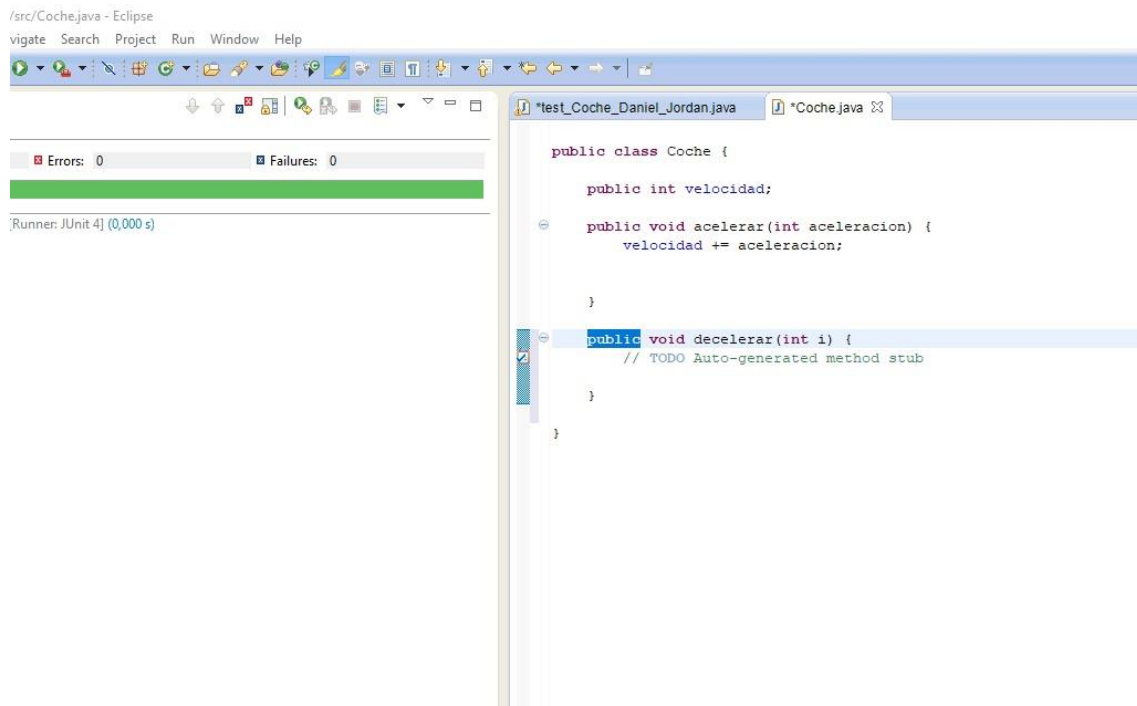


Ahora volvemos a copiar el método test anterior y lo pegamos, modificando el título del método por “al decelerar un coche su velocidad disminuye”

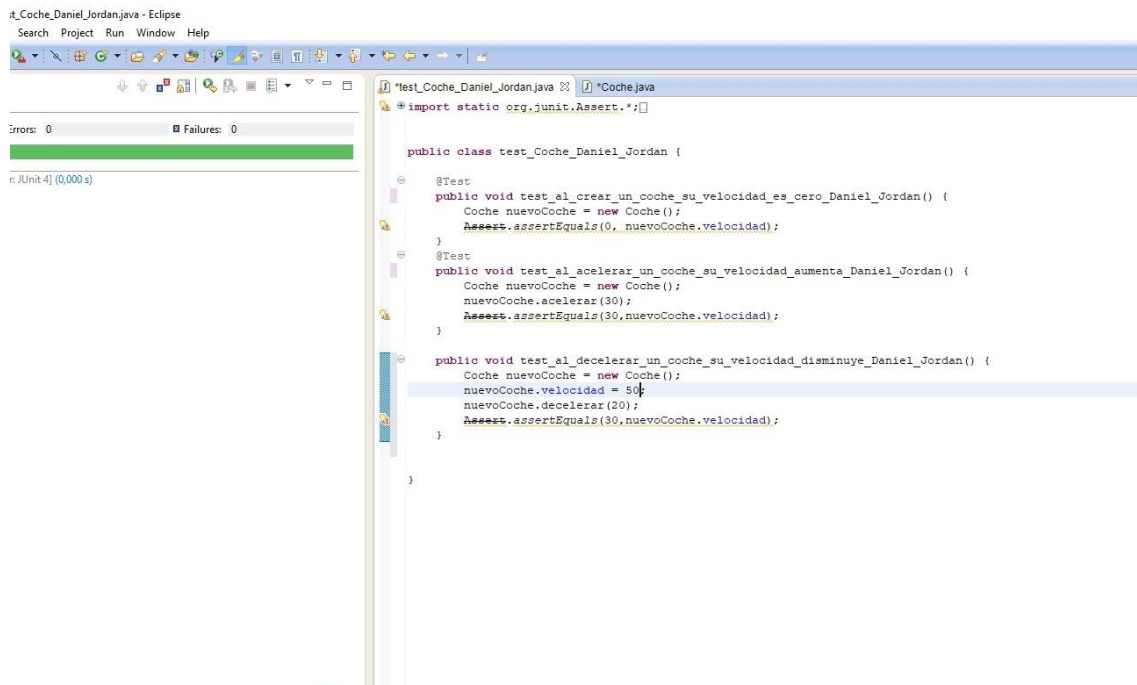


Ahora le pasamos el método decelerar el cual al no estar creado en la clase coche

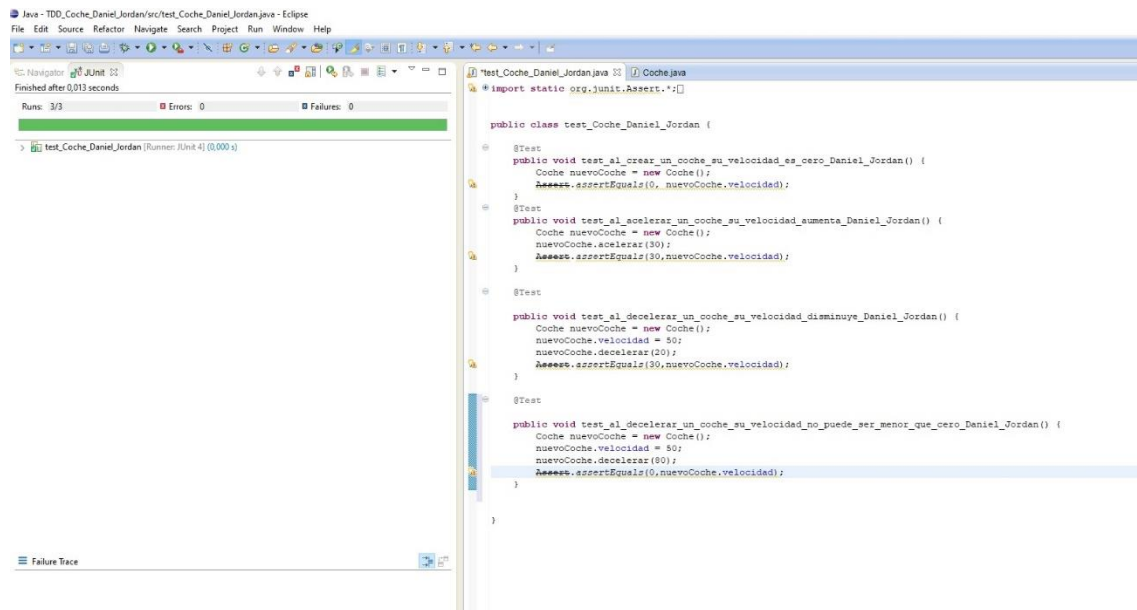
debemos hacer la misma operación que con el método acelerar. Clicamos a la crucecita roja que se nos crea en la misma línea en la cual le hemos pasado “nuevoCoche.decelerar” y nos invitará a crear el método decelerar en la clase Coche.



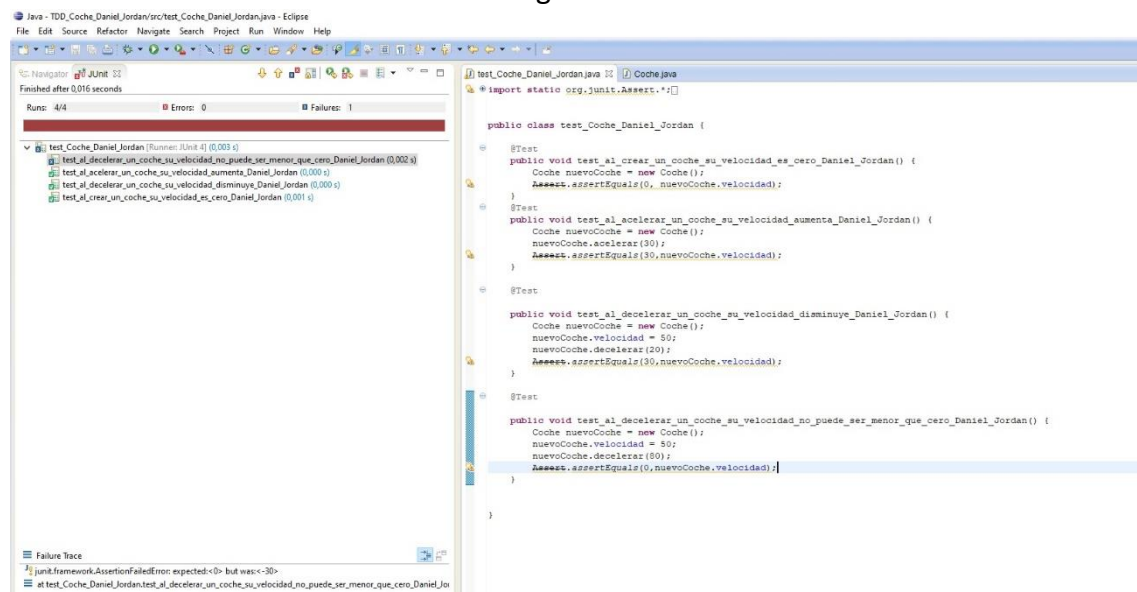
Nos lo crea con una variable de tipo entero y modificamos la “i” por deceleración. Dentro del método le decimos que “`velocidad -= deceleración;`” para que la velocidad resultante sea la resta de la deceleración que le pasamos al método.



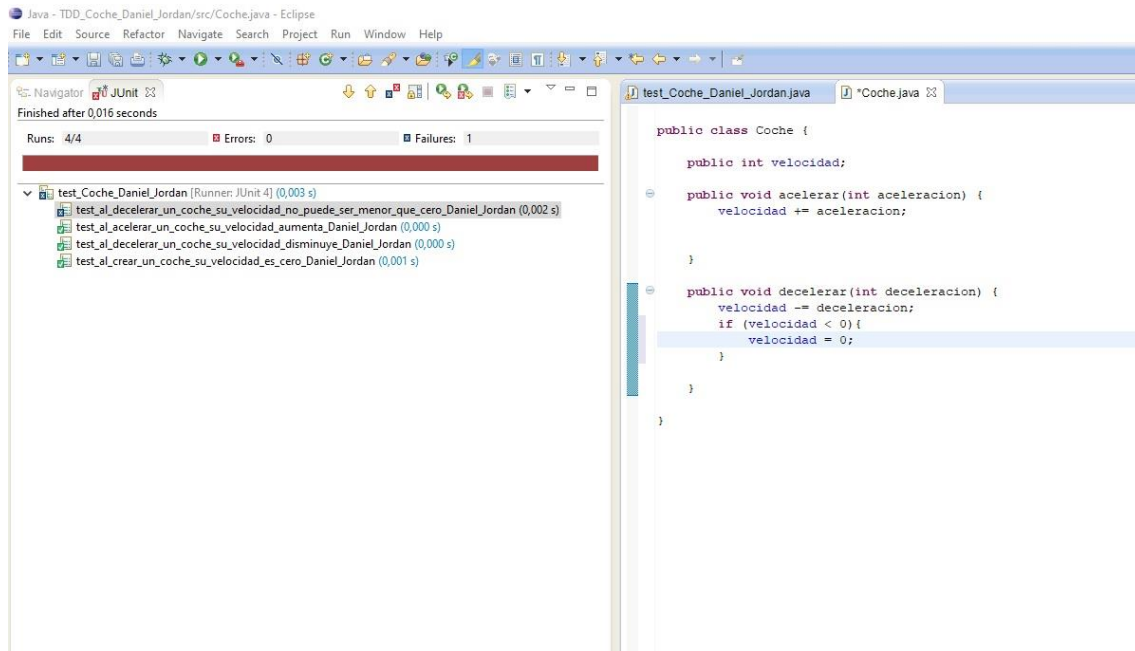
Vamos a darle una velocidad inicial de 50, y una deceleración de 80 y en el Assert le decimos que esperamos una velocidad de 0 resultante. Veremos que esto nos da un error.



Efectivamente en la siguiente imagen una vez que le damos al play verde del menú superior para probar el test, vemos que no funciona porque esperamos una velocidad de 0 pero como la deceleración es mayor que la velocidad del vehículo, nos la devuelve en número negativo menor a 0.



Para arreglar esto lo que tenemos que hacer es ir a la clase Coche y darle un condicional IF para indicarle que si la velocidad es menor a cero, esta pase a ser directamente 0.



En la siguiente imagen podemos ver como al ejecutar ahora el método podemos comprobar como ahora sí la devolución del método coincide con nuestras expectativas.

