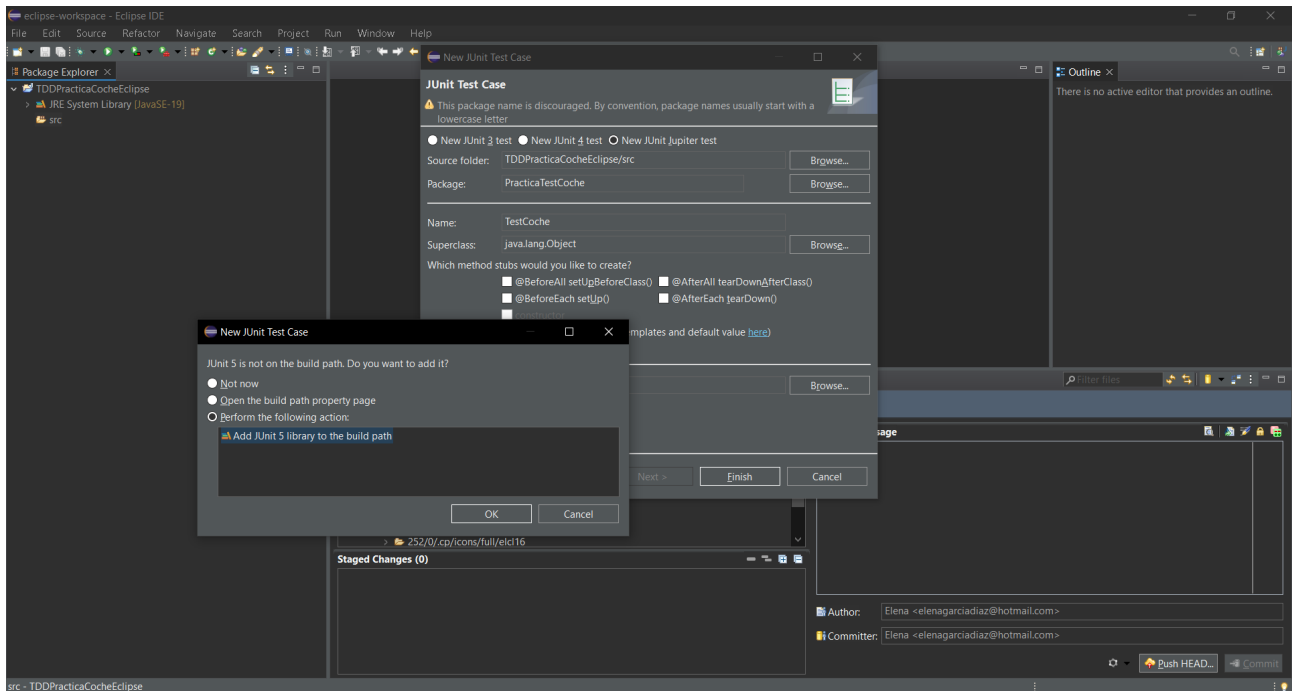


## Memoria: Mi Primer TDD V2.0 (con Eclipse)

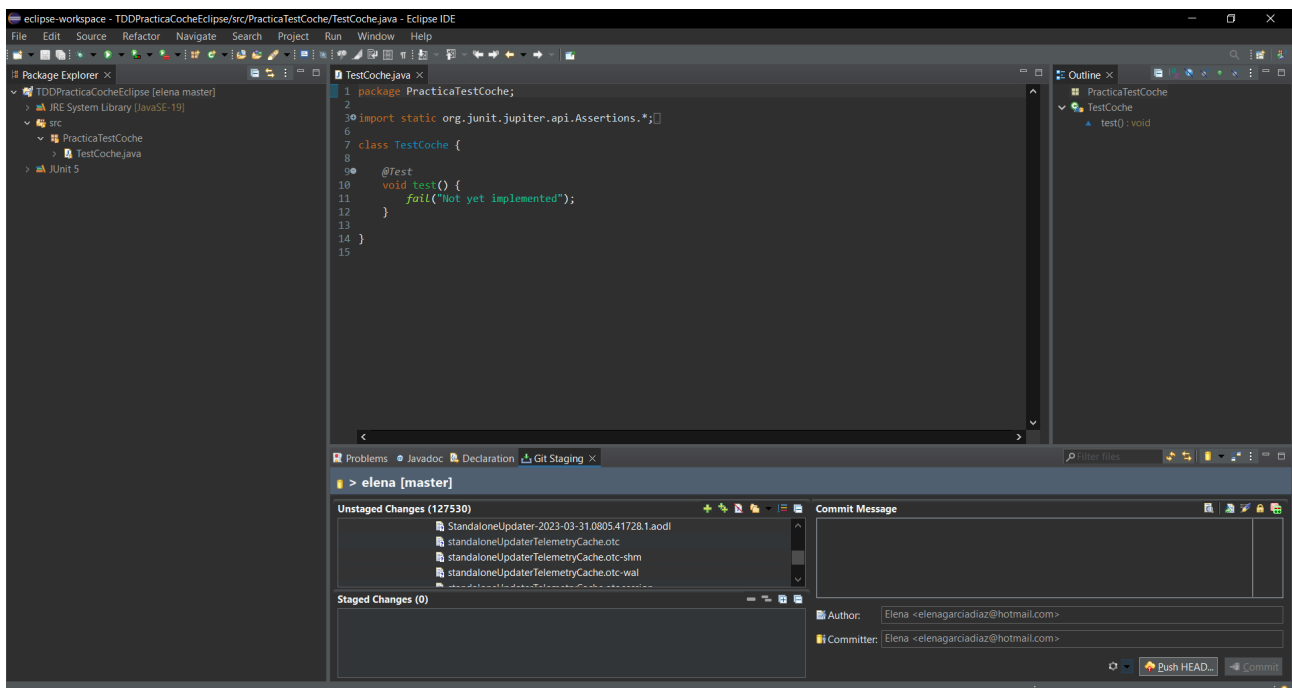
Para empezar a trabajar, abrimos Eclipse y creamos un nuevo proyecto java mediante: File-New-Java Project, al que llamaremos TDDPracticaCocheEclipse.

Hacemos click derecho sobre la carpeta src del proyecto y creamos una nueva clase JUnit mediante: New-JUnit Test Case, la llamaremos TestCoche. En este punto también le daremos un nombre al Package, ya que Eclipse nos lo pone por defecto, que será PracticaTestCoche. Marcaremos la opcion New JUnit Jupiter test para que nos importe las librerías de Junit5 al proyecto.

Sería así:



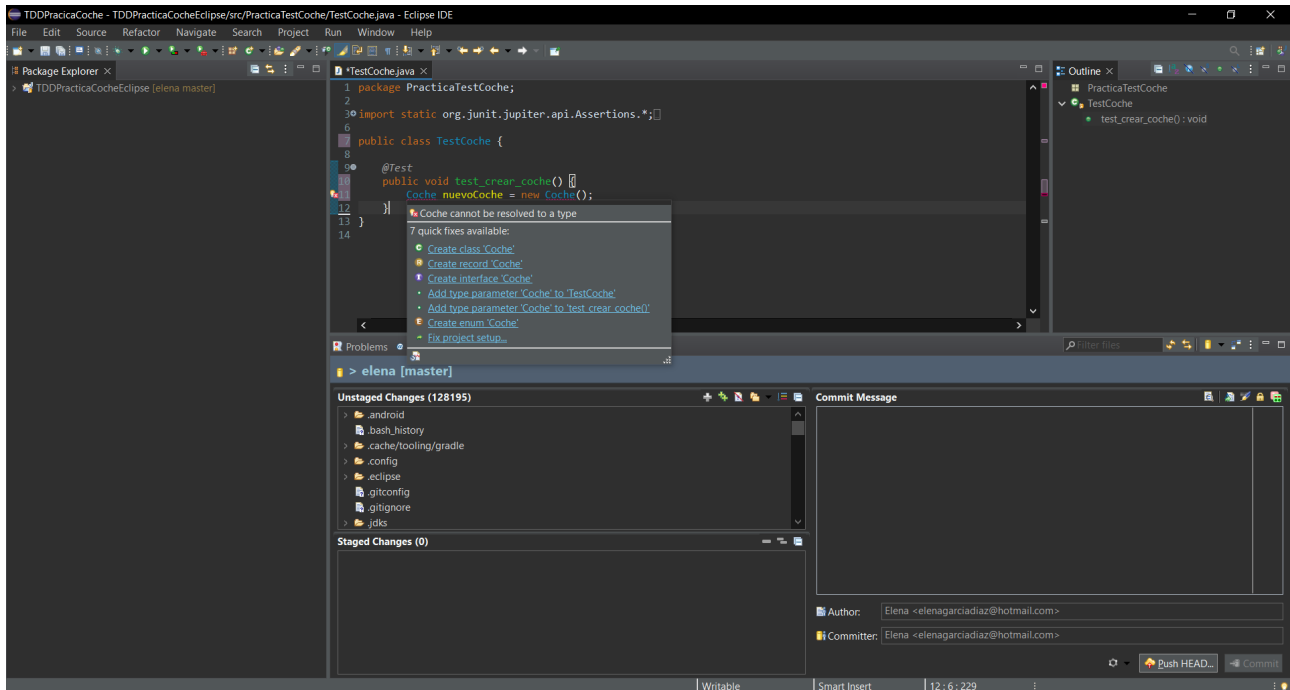
Y quedaría así:

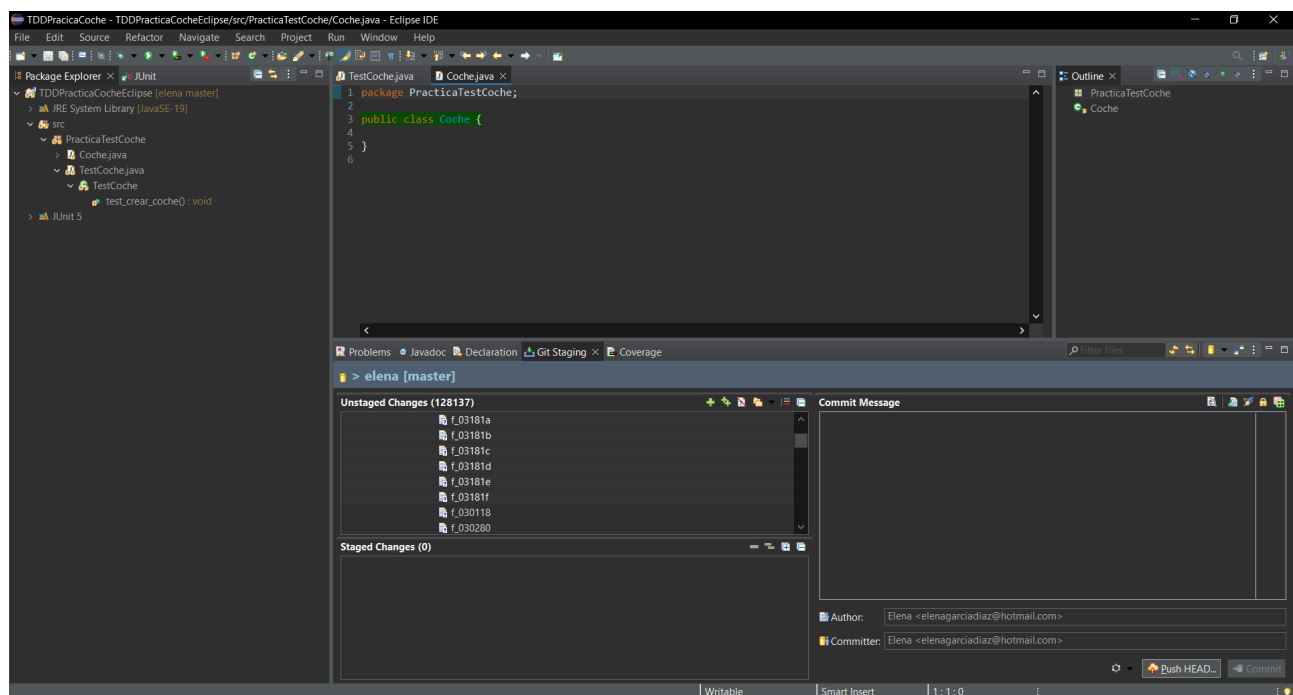


Ahora crearemos el primer test: En la clase TestCoche introducimos una línea de código para crear un objeto Coche de nombre nuevoCoche, dentro de test\_crear\_coche.

En este punto, podemos observar que al intentar crear el objeto nuevoCoche, Coche se nos marca en rojo. Nos está avisando de que no existe una clase Coche para poder crear objetos de ella, y que debemos crearla, así que nos ponemos sobre Coche con el cursor y hacemos click en Create class 'Coche'.

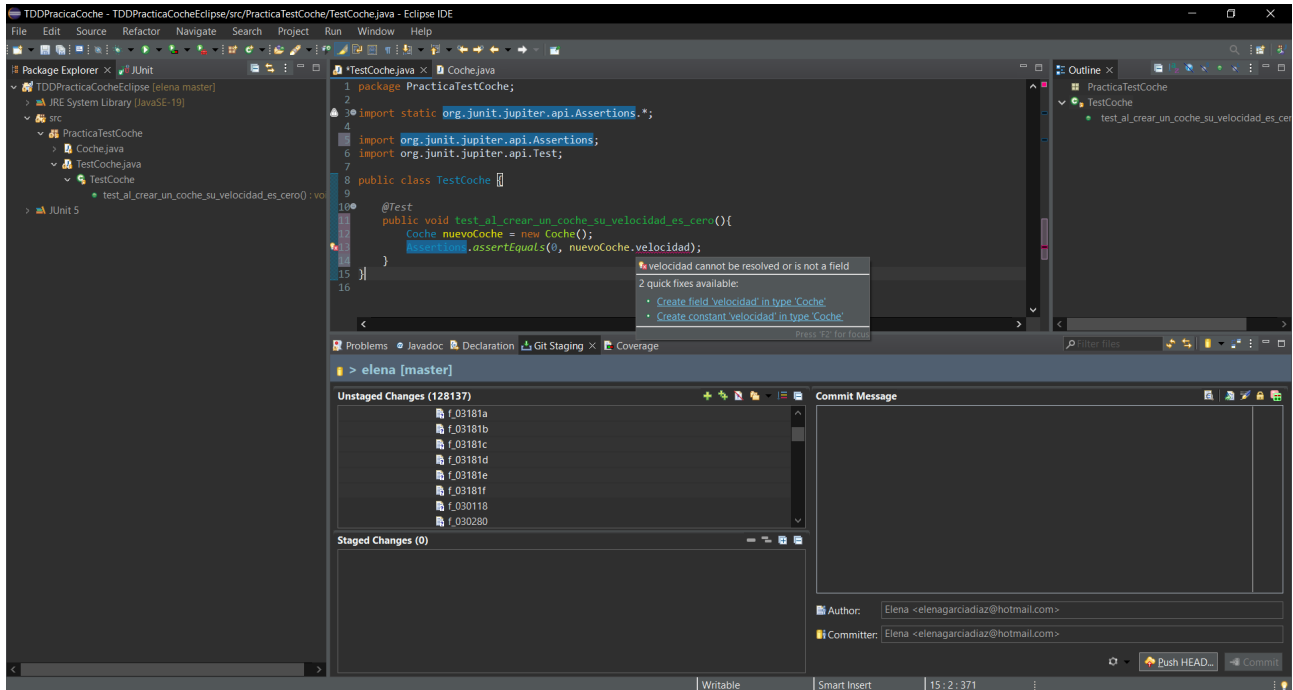
De esta forma:



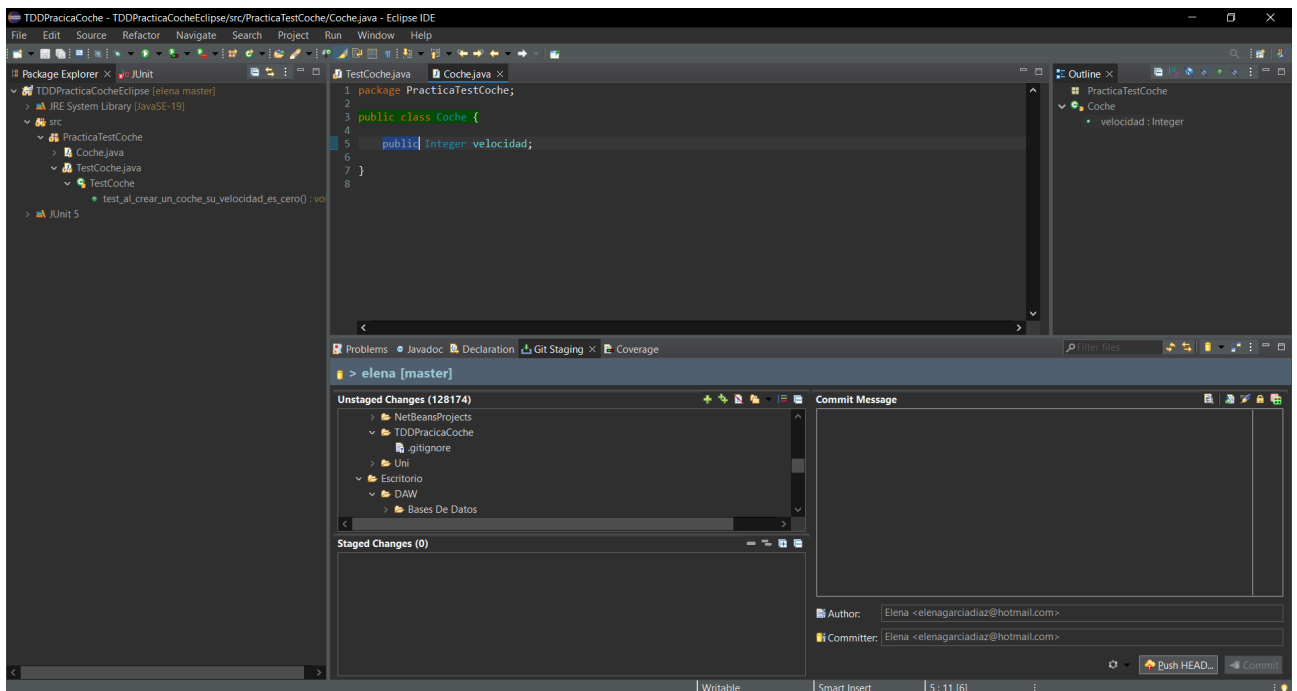
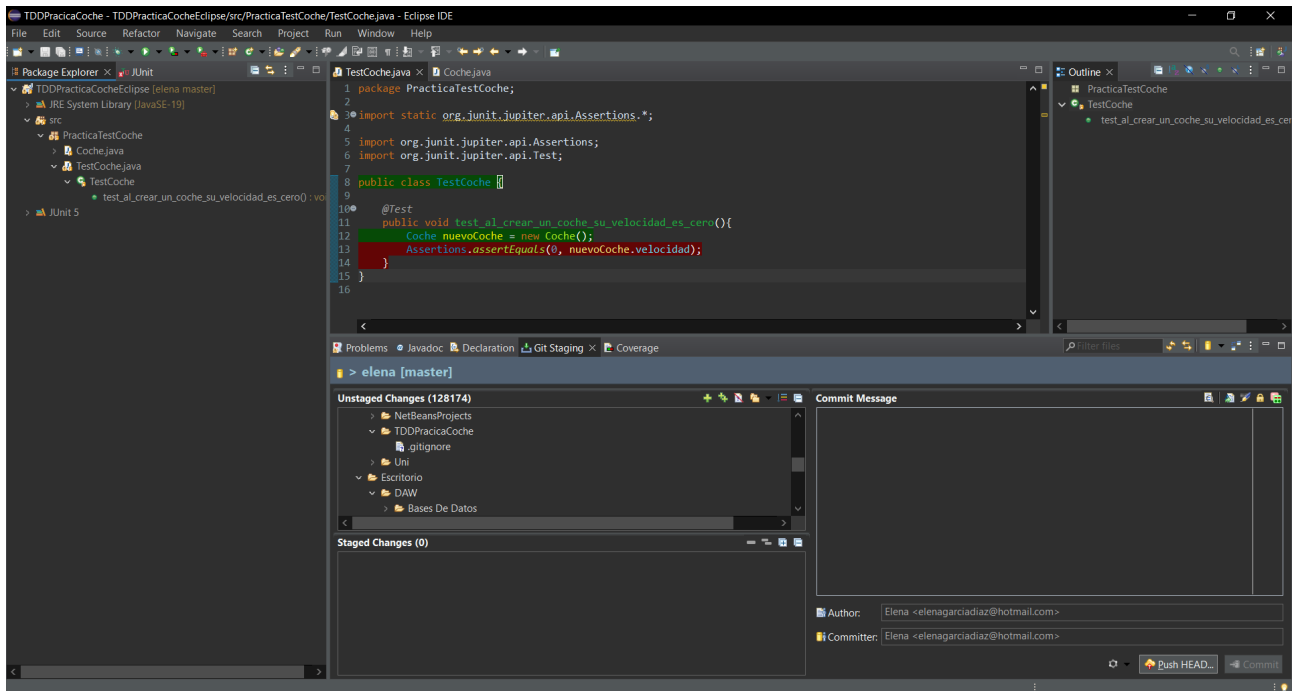


Ahora modificaremos el primer test para hacerlo un poco más complejo: Cambiamos el nombre del test a: `test_al_crear_un_coche_su_velocidad_es_cero`. En la clase `TestCoche` introducimos una línea de código para que la velocidad del objeto `nuevoCoche` sea 0, dentro de `test_al_crear_un_coche_su_velocidad_es_cero`. Nos pedirá que importemos las librerías `Assertions`. En esta ocasión nos marca en rojo `velocidad`. Nos avisa de que no existe la variable `velocidad` en la clase `Coche` y que debemos crearla, así que nos ponemos sobre `velocidad` con el cursor y hacemos click en `Create field 'velocidad' in type 'Coche'`.

De esta forma:



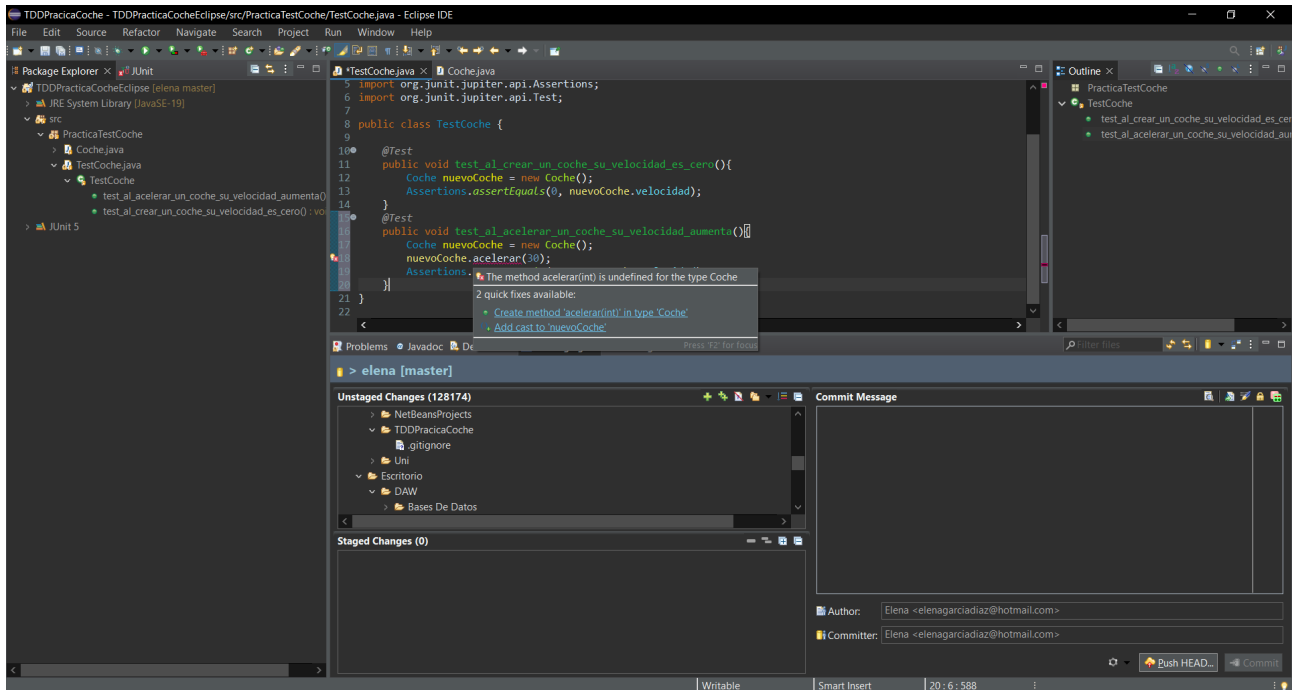
Le damos a Guardar, a Launch TestCoche y a Run TestCoche y el test queda pasado.  
Quedaría así:



Ahora crearemos el segundo test: En la clase TestCoche introducimos un nuevo test, al que llamaremos test\_al\_acelerar\_un\_coche\_su\_velocidad\_aumenta, y escribiremos una línea de código para llamar al método acelerar.

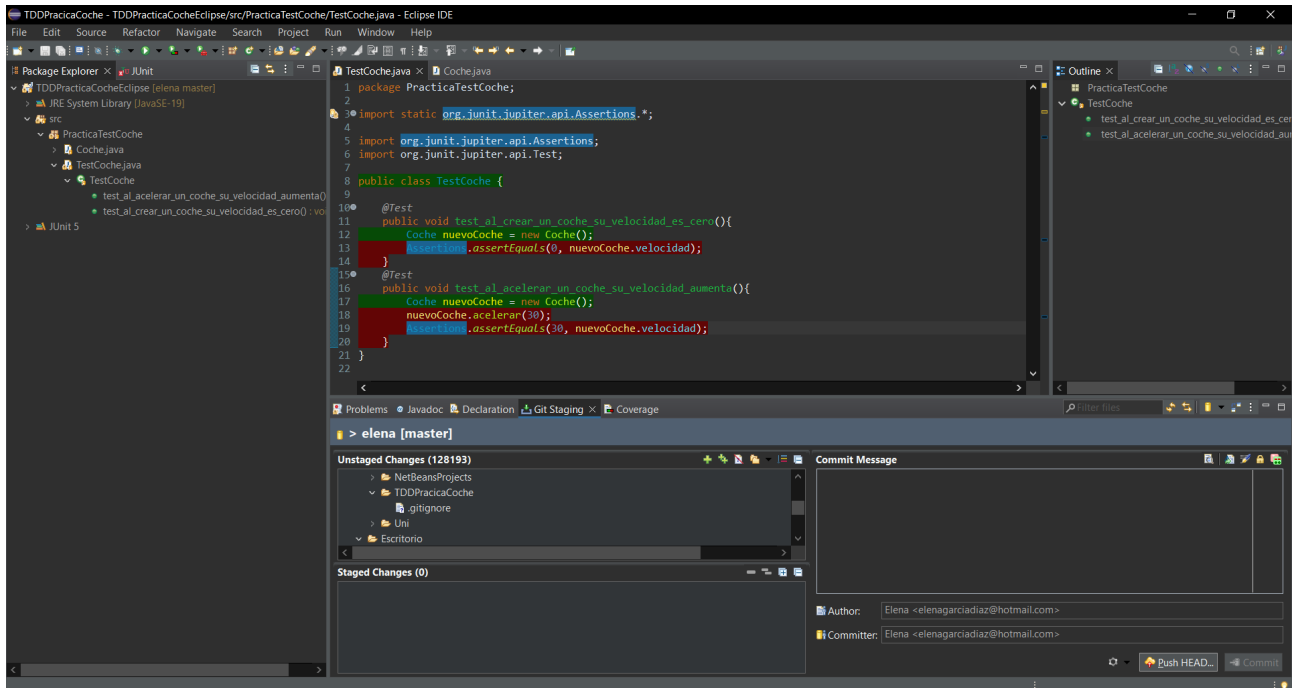
En esta ocasión nos marca en rojo acelerar. Nos avisa de que no existe el método acelerar en la clase Coche y que debemos crearlo, así que nos ponemos sobre acelerar con el cursor y hacemos click en Create method 'acelerar(int)' in type 'Coche'.

De esta forma:

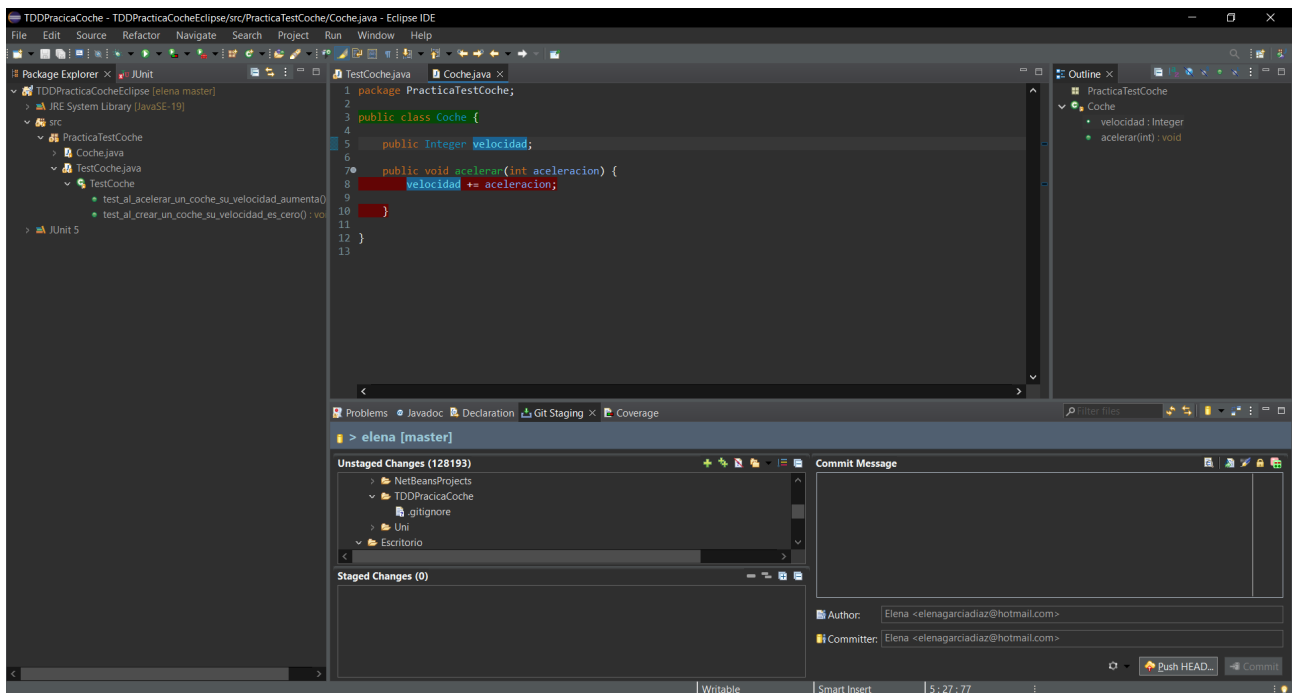


Como hemos añadido un método a la clase Coche, escribimos el código con lo que hará, en este caso a la velocidad se le sumará la variable aceleracion.

Le damos a Guardar, a Launch TestCoche y a Run TestCoche y el test queda pasado. Quedaría así:



```
1 package PracticaTestCoche;
2
3 import static org.junit.jupiter.api.Assertions.*;
4 import org.junit.jupiter.api.Assertions;
5 import org.junit.jupiter.api.Test;
6
7 public class TestCoche {
8
9     @Test
10    public void test_al_crear_un_coche_su_velocidad_es_cero(){
11        Coche nuevoCoche = new Coche();
12        Assertions.assertEquals(0, nuevoCoche.velocidad);
13    }
14
15    @Test
16    public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
17        Coche nuevoCoche = new Coche();
18        nuevoCoche.acelerar(30);
19        Assertions.assertEquals(30, nuevoCoche.velocidad);
20    }
21 }
22
```

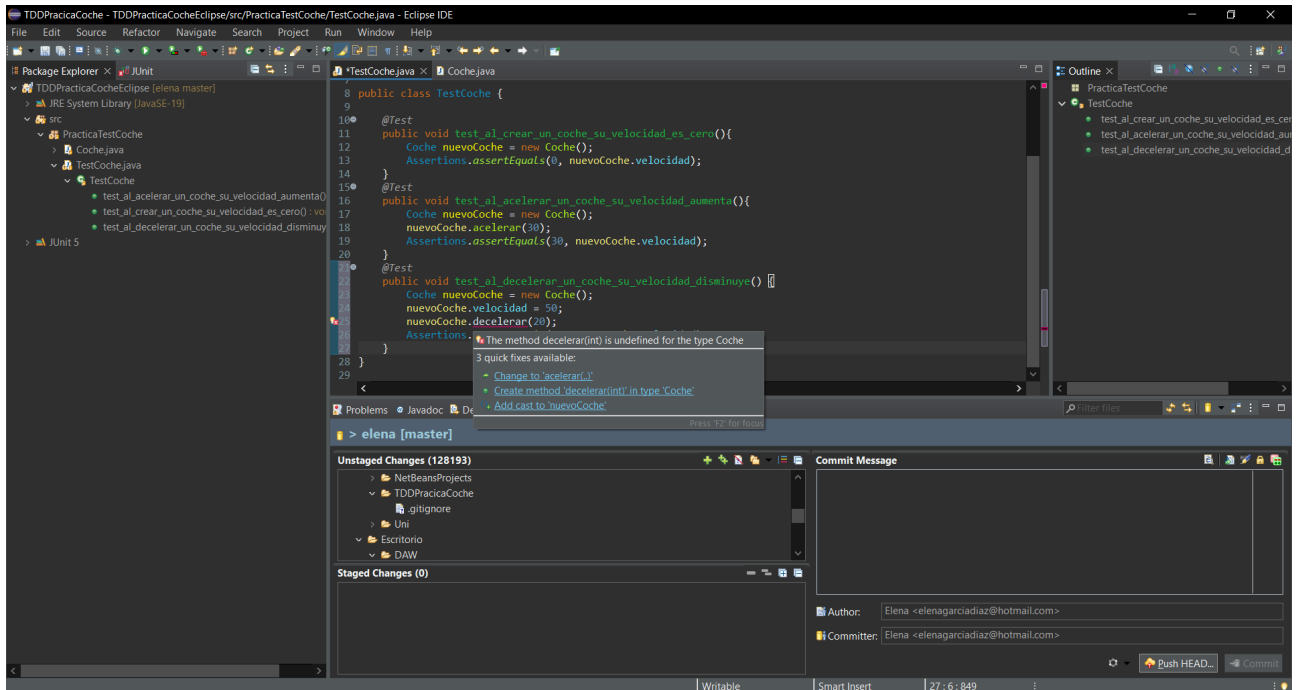


```
1 package PracticaTestCoche;
2
3 public class Coche {
4
5     public Integer velocidad;
6
7     public void acelerar(int aceleracion) {
8         velocidad += aceleracion;
9     }
10 }
11
12
13
```

Ahora crearemos el tercer test: En la clase TestCoche introducimos un nuevo test, al que llamaremos `test_al_decelerar_un_coche_su_velocidad_disminuye`, y escribiremos una línea de código para llamar al método `decelerar`.

En esta ocasión nos marca en rojo `decelerar`. Nos avisa de que no existe el método `decelerar` en la clase `Coche` y que debemos crearlo, así que nos ponemos sobre `decelerar` con el cursor y hacemos click en `Create method 'decelerar(int)' in type 'Coche'`.

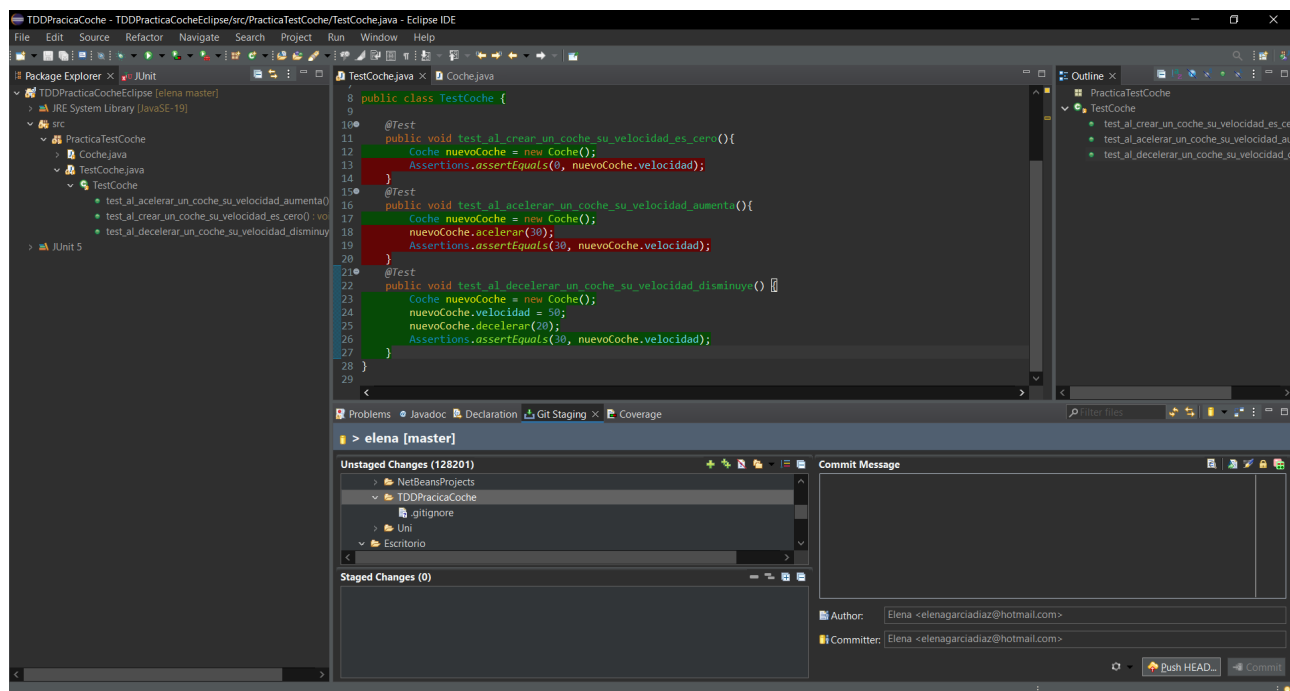
De esta forma:





Igual que en el test anterior, hemos añadido un método a la clase Coche, así que escribimos el código con lo que hará, y en este caso a la velocidad se le restará la variable deceleracion.

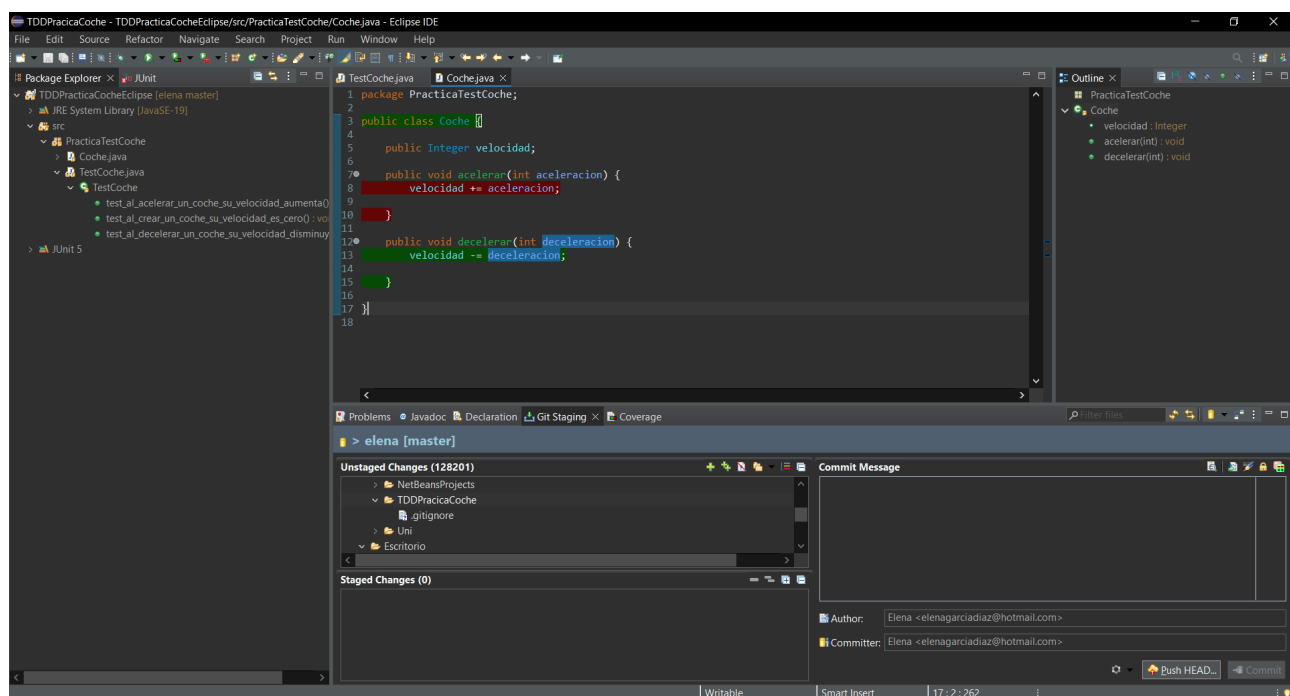
Le damos a Guardar, a Launch TestCoche y a Run TestCoche y el test queda pasado. Quedaría así:



The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows the project structure with `PracticaTestCoche` containing `Coche.java` and `TestCoche.java`.
- Editor:** Displays `TestCoche.java` with the following code:

```
8 public class TestCoche {
9
10     @Test
11     public void test_al_crear_un_coche_su_velocidad_es_cero(){
12         Coche nuevoCoche = new Coche();
13         Assertions.assertEquals(0, nuevoCoche.velocidad);
14     }
15
16     @Test
17     public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
18         Coche nuevoCoche = new Coche();
19         nuevoCoche.acelerar(30);
20         Assertions.assertEquals(30, nuevoCoche.velocidad);
21     }
22
23     @Test
24     public void test_al_decelerar_un_coche_su_velocidad_disminuye(){
25         Coche nuevoCoche = new Coche();
26         nuevoCoche.velocidad = 50;
27         nuevoCoche.decelerar(20);
28         Assertions.assertEquals(30, nuevoCoche.velocidad);
29     }
30 }
```
- Git Staging:** Shows uncommitted changes for `TestCoche.java`.
- Commit Message:** A text area for entering a commit message.
- Author/Committer:** Fields for `Elena <elenagarciadiaz@hotmail.com>`.
- Buttons:** `Push HEAD...` and `Commit`.

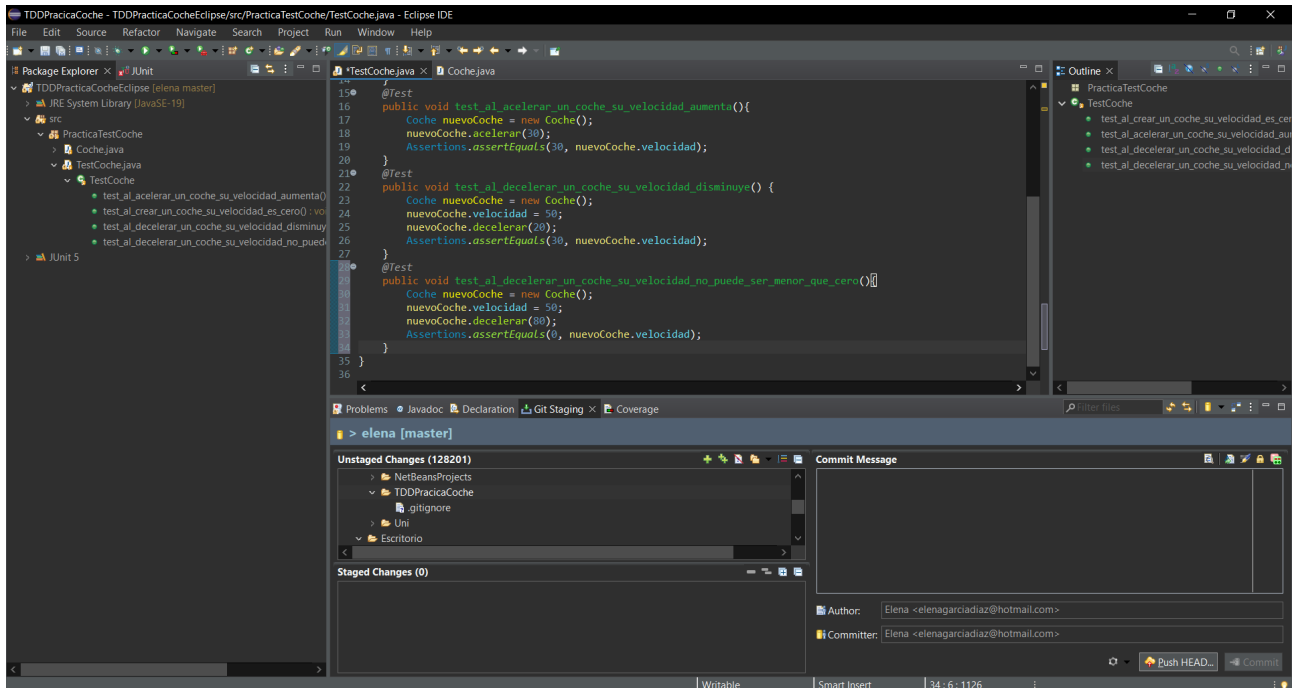


The screenshot shows the Eclipse IDE with the following components:

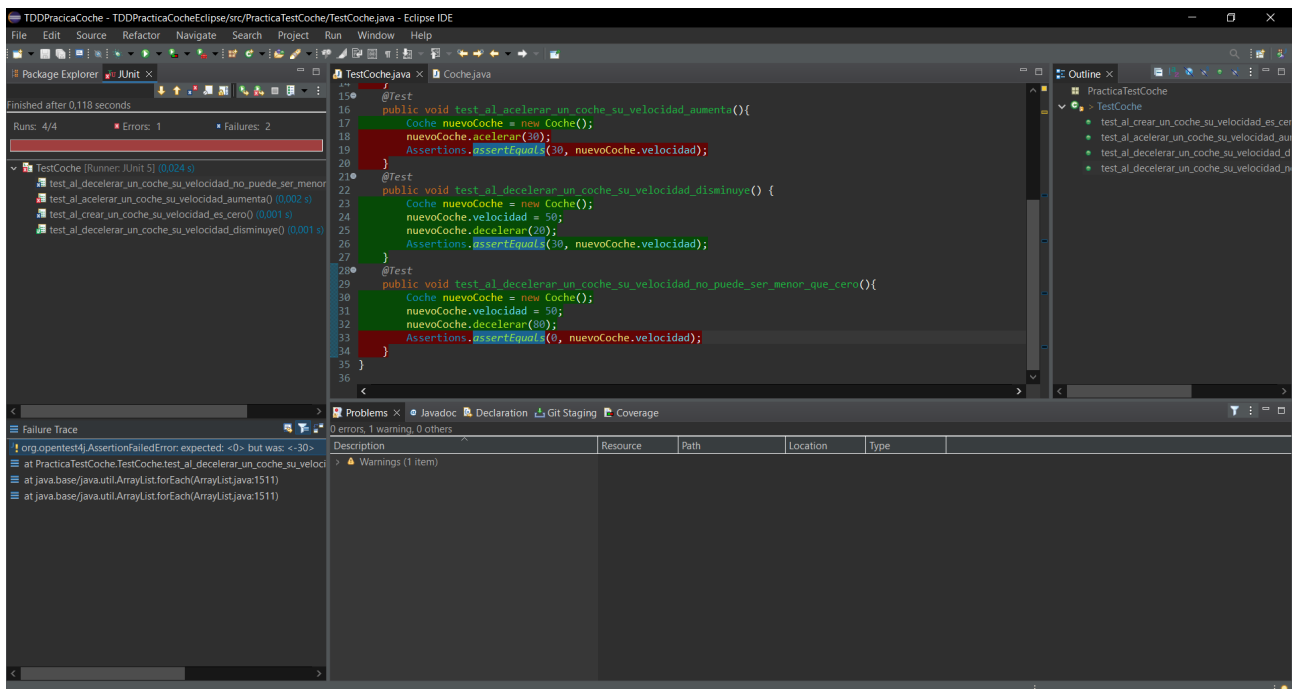
- Package Explorer:** Shows the project structure with `PracticaTestCoche` containing `Coche.java` and `TestCoche.java`.
- Editor:** Displays `Coche.java` with the following code:

```
1 package PracticaTestCoche;
2
3 public class Coche {
4
5     public Integer velocidad;
6
7     public void acelerar(int aceleracion) {
8         velocidad += aceleracion;
9     }
10
11
12     public void decelerar(int deceleracion) {
13         velocidad -= deceleracion;
14     }
15
16 }
17
18 }
```
- Git Staging:** Shows uncommitted changes for `Coche.java`.
- Commit Message:** A text area for entering a commit message.
- Author/Committer:** Fields for `Elena <elenagarciadiaz@hotmail.com>`.
- Buttons:** `Push HEAD...` and `Commit`.

Ahora crearemos el cuarto y último test: En la clase TestCoche introducimos un nuevo test, al que llamaremos `test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero`, y escribiremos una línea de código para evitar que, al restarle la deceleración a la velocidad, nos quede un resultado en negativo. Aparentemente, no hay ningún problema y puede compilar perfectamente, como podemos ver aquí:



Pero al ejecutarlo, ocurre lo siguiente:



En la parte de abajo a la izquierda, marcado en azul, nos avisa de que, a pesar de que el resultado esperado que hemos especificado en el test es 0, el resultado real es -30. Y con esta información ya podemos solucionar el problema, en este caso escribiendo una línea de código en el método decelerar para que, cuando la velocidad pretenda ser menor que 0, la iguale a 0.

Ahora si, le damos a Guardar, a Launch TestCoche y a Run TestCoche y el test queda pasado.  
Quedaría así:

