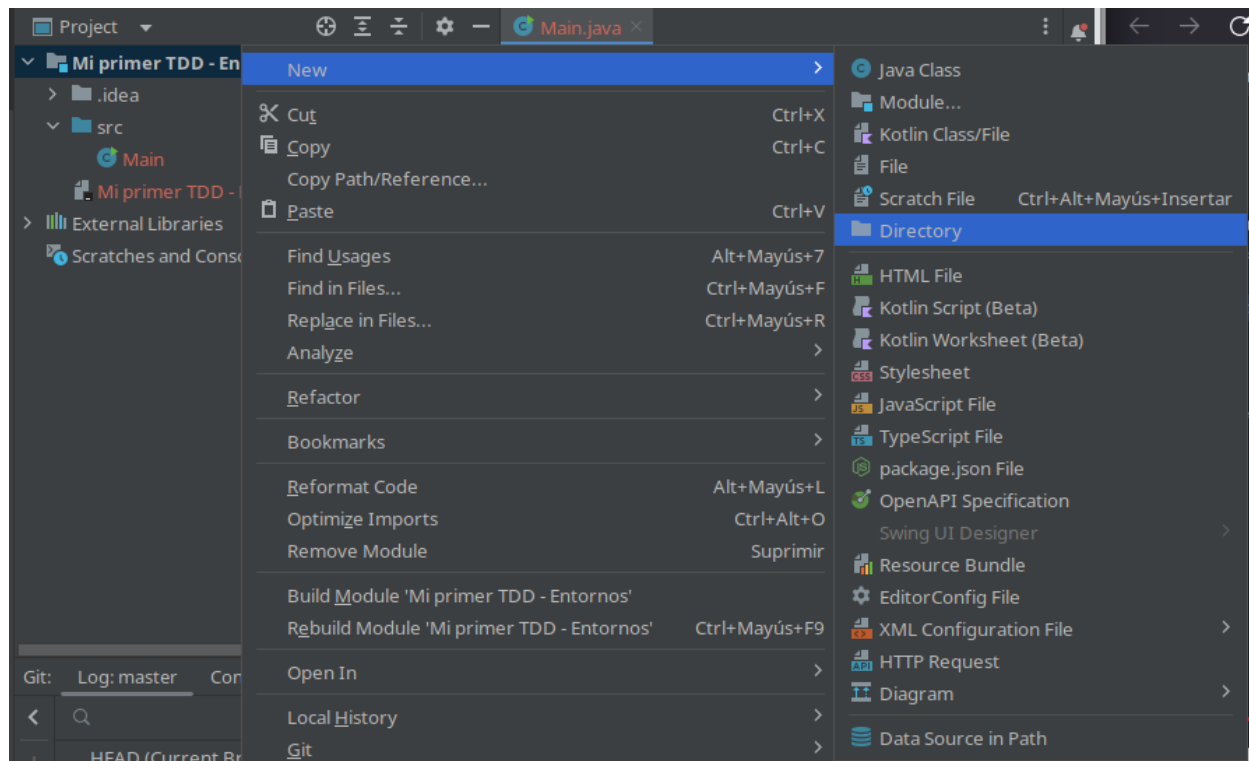
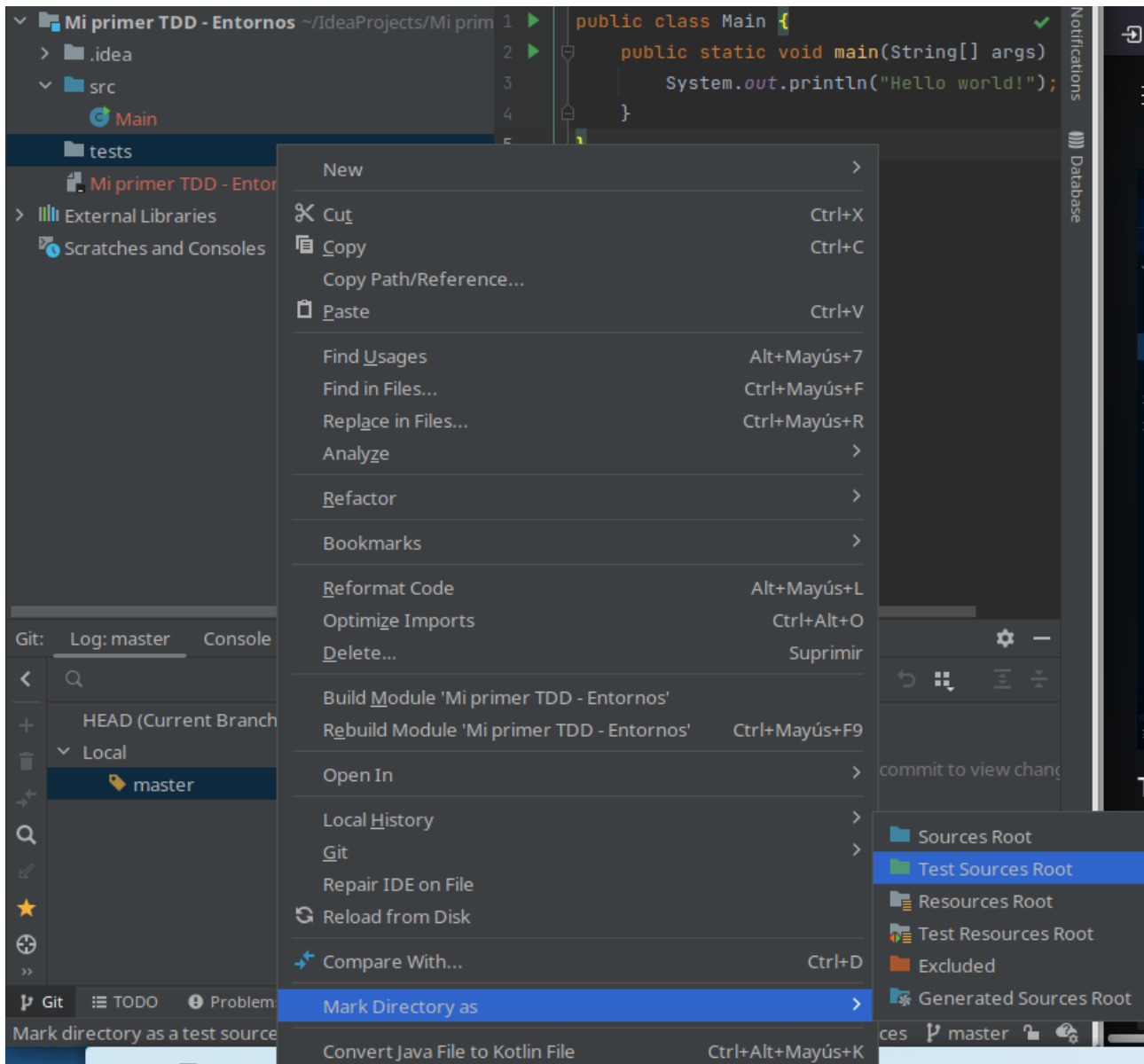


MI PRIMER TDD – ÁNGEL MACIÁ GARCÍA

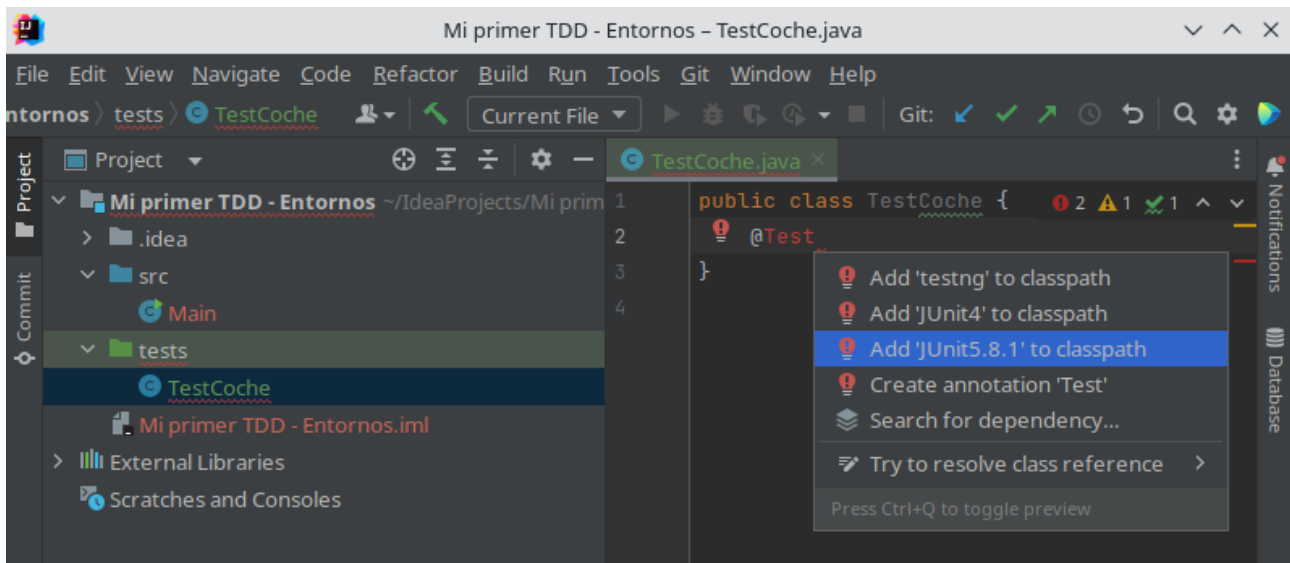
Paso 1: Lo primero que vamos a hacer es crear un nuevo proyecto de Java en IntelliJ. Una vez creado, pulsamos clic derecho en la carpeta del proyecto y vamos a la opción de “New > Directory”, le ponemos el nombre de “tests”, ya que este directorio lo utilizaremos para hacer tests.



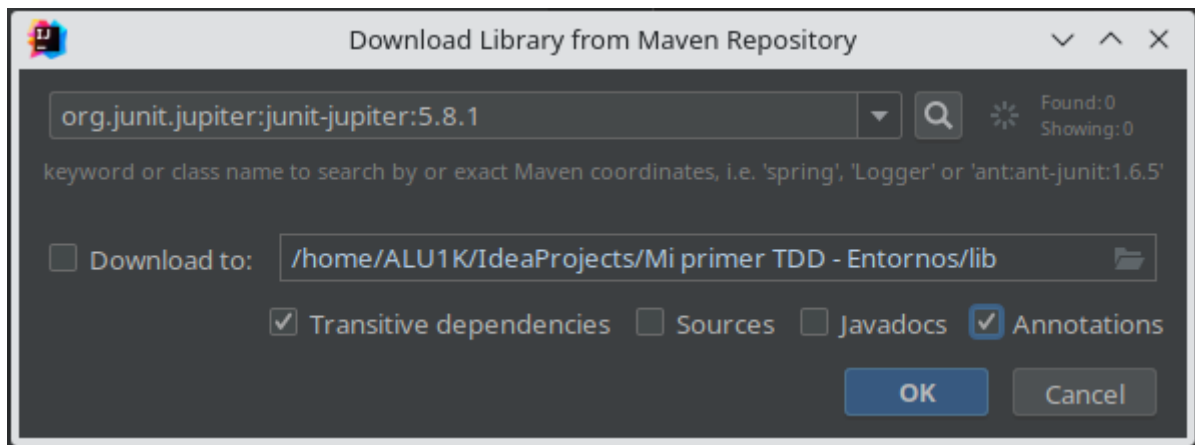
Paso 2: Ahora que hemos creado el directorio con el nombre "tests", debemos marcarlo como "Test Sources Root". Para ello, pulsamos clic derecho sobre "test" y seleccionamos "Mark Directory as > Test Sources Root".



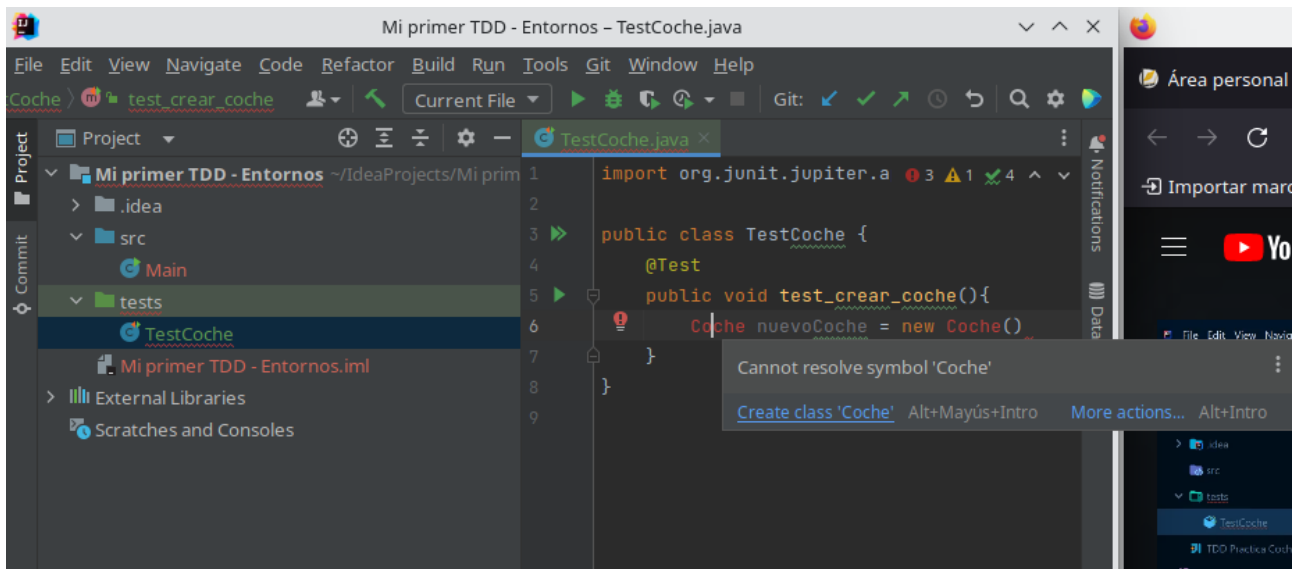
Paso 3: Ahora, dentro del directorio creamos una nueva clase Java pulsando "clic derecho en tests > New > Java Class". Llamaremos a la clase "TestCoche" y dentro de ella indicamos que es un test escribiendo "@Test", seguido IntelliJ nos indica que no sabe qué es "@Test", por lo que si hacemos clic en él, nos da opciones para solucionarlo, debemos pulsar en "Add JUnit5.8.1 to classpath".



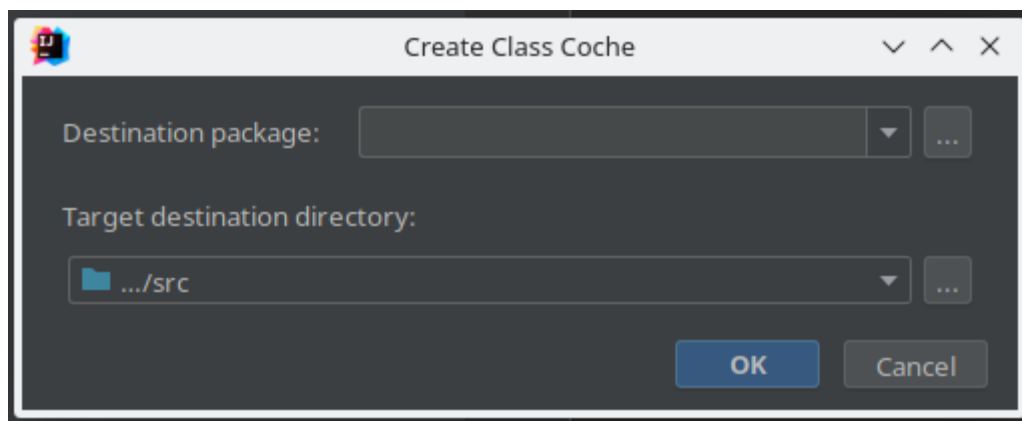
Después de haber pulsado la opción indicada anteriormente, se abre una ventanita en la que debemos marcar la opción "Annotations", que de por sí no viene marcada y pulsamos "OK".



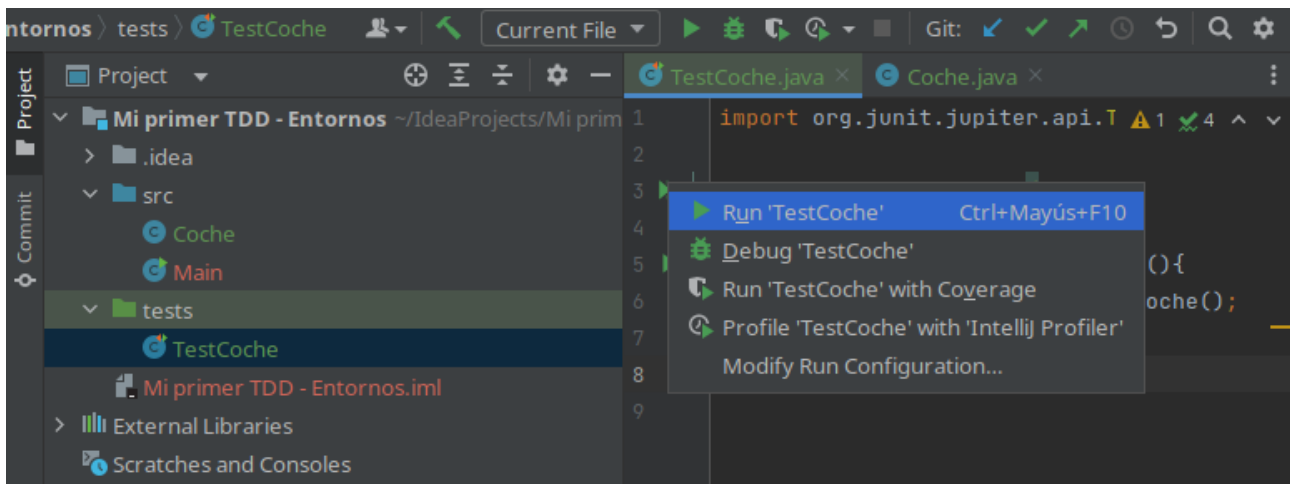
Paso 4: Vamos a crear nuestro primer test, se llamará "test_crear_coche", dentro de este test vamos a crear un objeto llamado "nuevoCoche" de la clase "Coche". Al indicar esto, como la clase "Coche" no existe, se muestra en rojo, pero IntelliJ nos propone crear la clase "Coche", debemos pulsamos en esa opción.



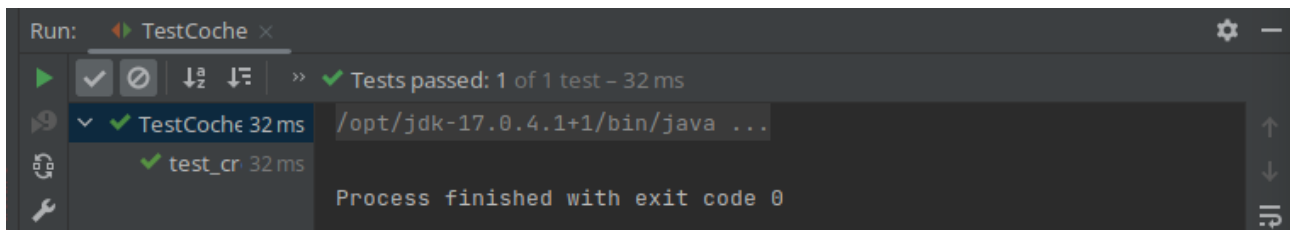
Cuando indicamos que cree la clase Coche, nos pregunta sobre su ubicación, dejamos la predeterminada y pulsamos "OK".



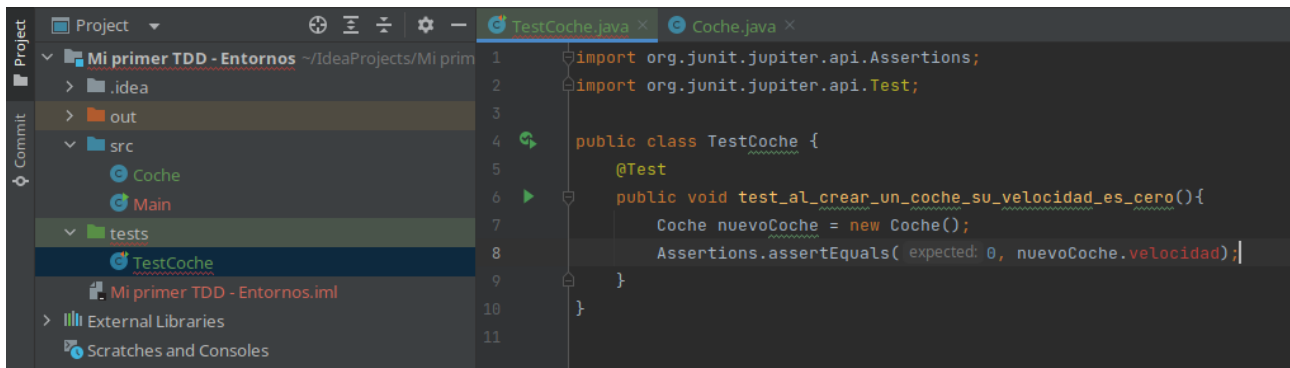
Paso 5: Ahora ejecutamos el Test pulsando en el botón de "Play" y seleccionando la opción "Run TestCoche".



Cuando acaba de ejecutarse, vemos que el Test no ha dado ningún problema, esto también es debido a que es muy simple.

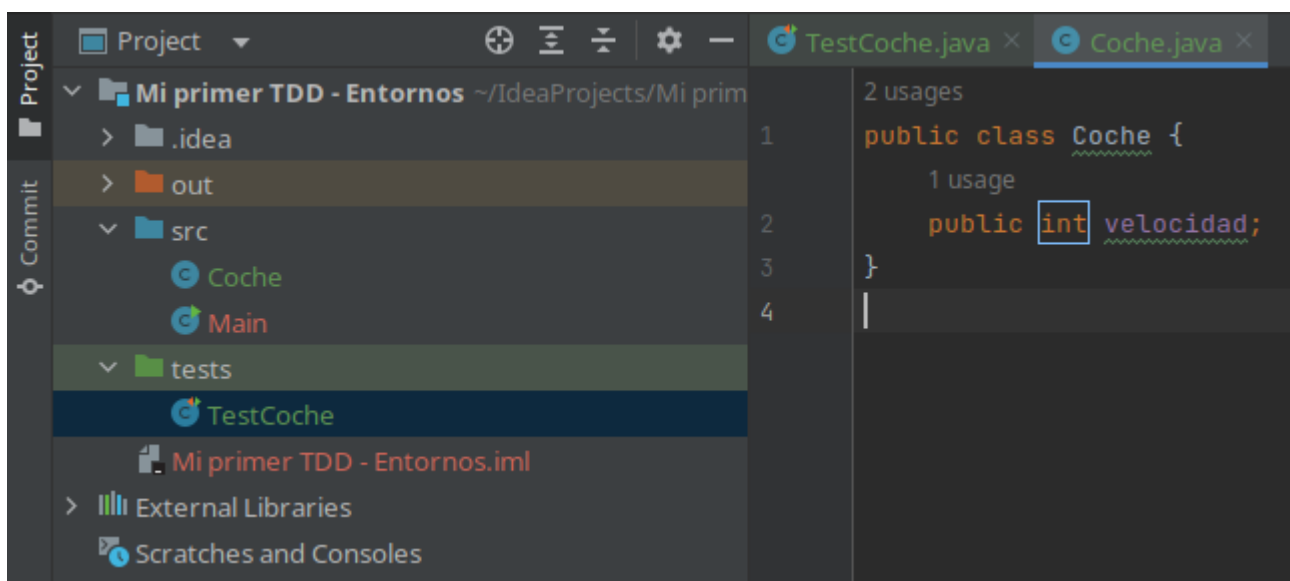


Paso 6: Ahora modificamos el test creado, empezando por su nombre, lo cambiamos a "test_al_crear_un_coche_su_velocidad_es_cero". Después de cambiar el nombre, dentro del test añadimos un "Assertions.assertEquals(0, nuevoCoche.velocidad)". Esto nos servirá para comprobar que la velocidad del "nuevoCoche" tiene un valor de 0.



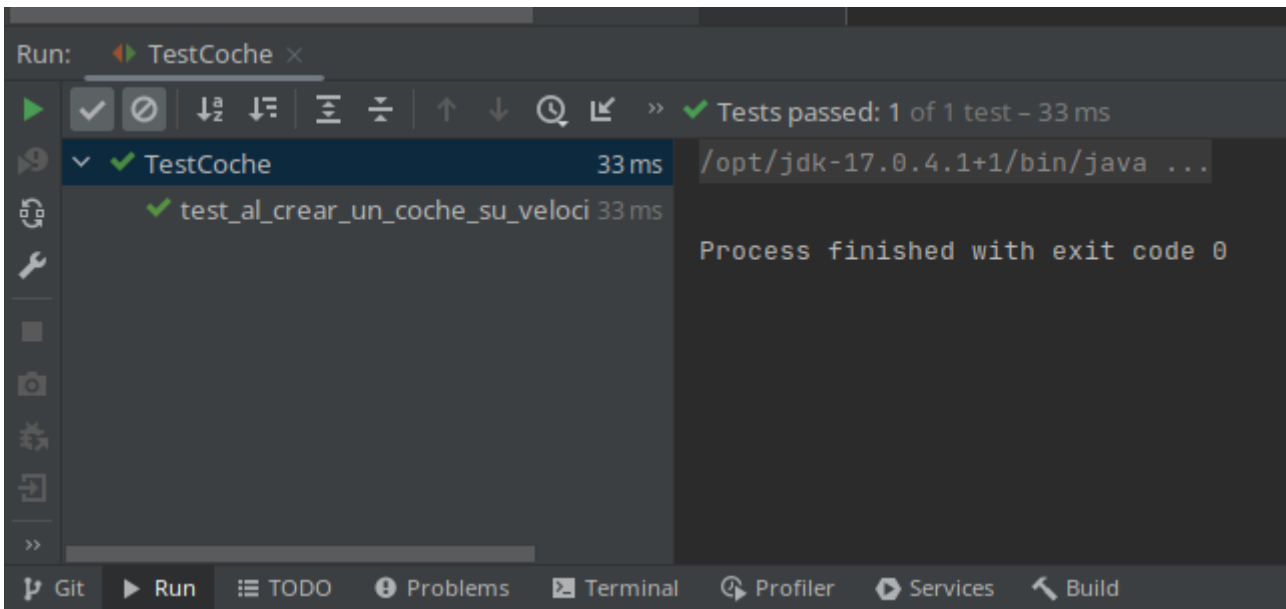
```
1 import org.junit.jupiter.api.Assertions;
2 import org.junit.jupiter.api.Test;
3
4 public class TestCoche {
5     @Test
6     public void test_al_crear_un_coche_su_velocidad_es_cero(){
7         Coche nuevoCoche = new Coche();
8         Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
9     }
10 }
11
```

Ya que el atributo "velocidad" no existe, IntelliJ nos lo indica y nos da la opción de crearlo, igual que antes con la clase "Coche", pulsamos en "Create field velocidad in Coche". Por defecto el atributo velocidad lo pone como un entero, pero eso está bien así que no lo cambiamos.

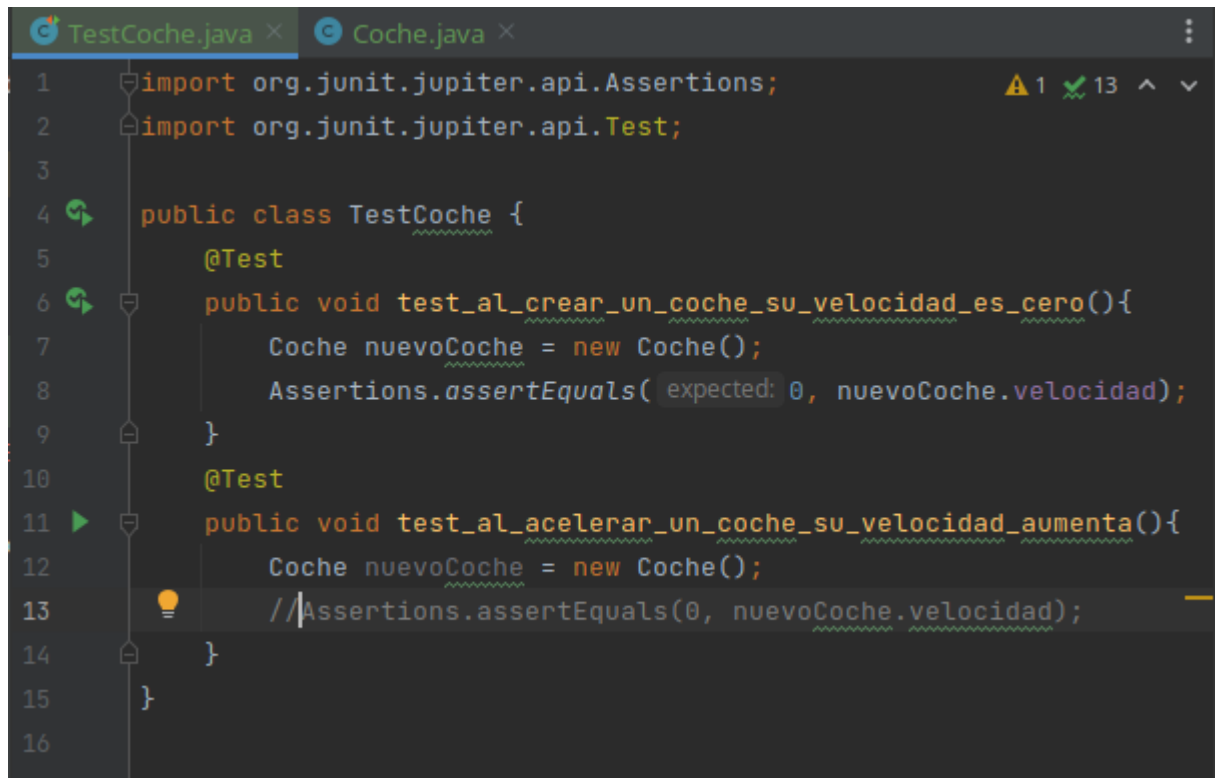


```
1 public class Coche {
2     public int velocidad;
3 }
4
```

Paso 7: Vamos a comprobar de nuevo que todo está bien, ejecutamos el Test y vemos que no ha habido ningún problema.

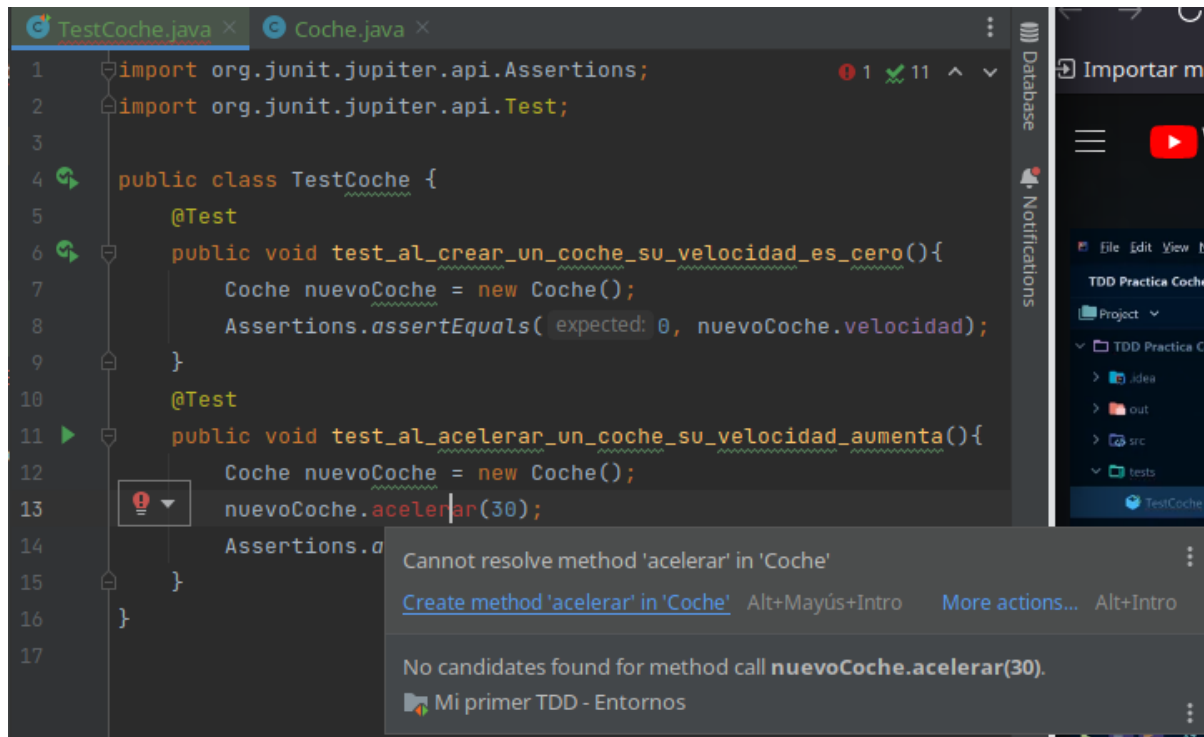


Paso 8: Creamos un nuevo Test llamado "test_al_acelerar_un_coche_su_velocidad_aumenta". Dentro de este test crearemos el objeto "nuevoCoche" de la clase Coche y también hacemos un "assertEquals" igual que el de antes, como no lo vamos a utilizar de momento, lo comentamos.



```
1  import org.junit.jupiter.api.Assertions;
2  import org.junit.jupiter.api.Test;
3
4  public class TestCoche {
5      @Test
6      public void test_al_crear_un_coche_su_velocidad_es_cero(){
7          Coche nuevoCoche = new Coche();
8          Assertions.assertEquals(0, nuevoCoche.velocidad);
9      }
10     @Test
11     public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
12         Coche nuevoCoche = new Coche();
13         //Assertions.assertEquals(0, nuevoCoche.velocidad);
14     }
15 }
16
```


Paso 9: En este Test vamos a utilizar el método "acelerar" en el objeto "nuevoCoche", además por parámetro vamos a introducir un 30, que será la velocidad que acelere el coche. De nuevo IntelliJ nos indica que "acelerar" no existe, pulsamos en "Create method acelerar in Coche". El "assertEquals" que teníamos comentado, lo modificamos poniendo que el valor esperado de "nuevoCoche.velocidad" es "30" y descomentamos la línea.



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

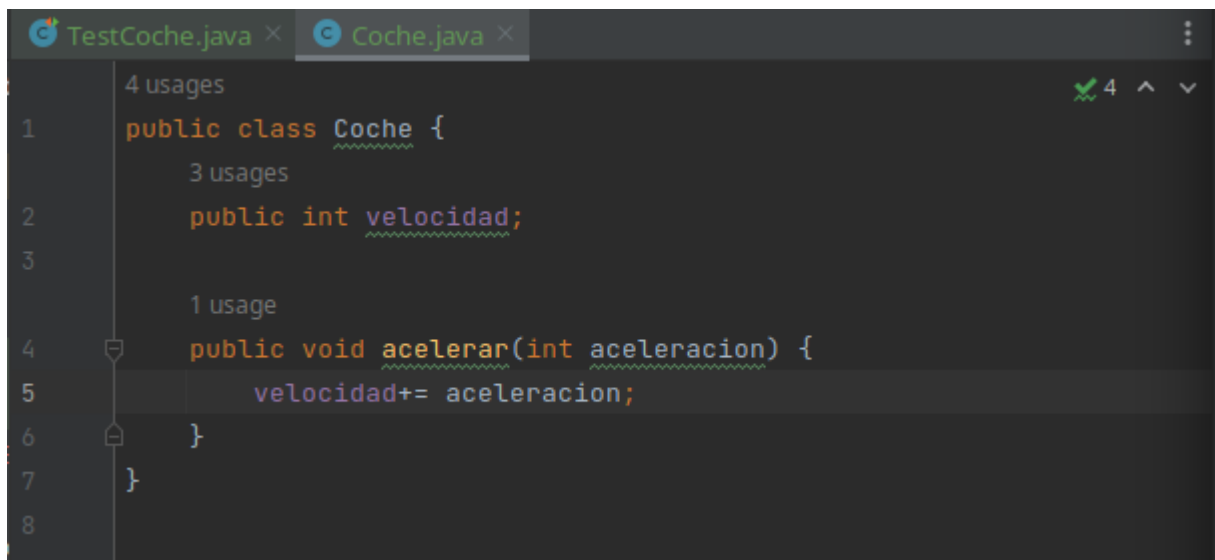
Cannot resolve method 'acelerar' in 'Coche'

Create method 'acelerar' in 'Coche' Alt+Mayús+Intro More actions... Alt+Intro

No candidates found for method call nuevoCoche.acelerar(30).

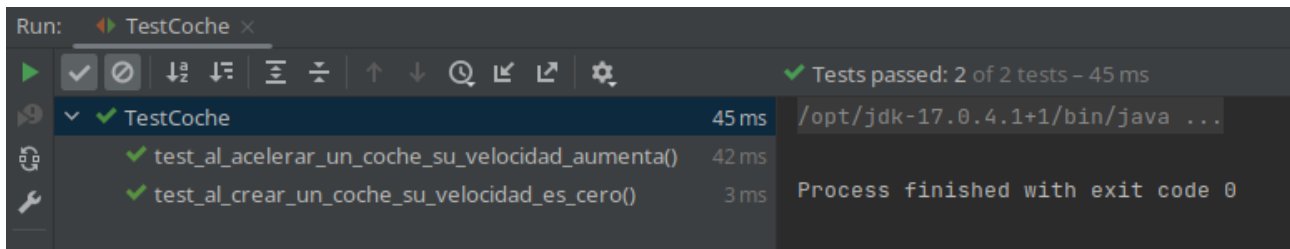
Mi primer TDD - Entornos

Vamos a la clase "Coche" y comprobamos que el método se ha creado. El método se crea con un parámetro "int i", este parámetro lo cambiamos y ponemos "int aceleracion". Dentro de este método haremos que la velocidad sea igual a la velocidad más la aceleración.

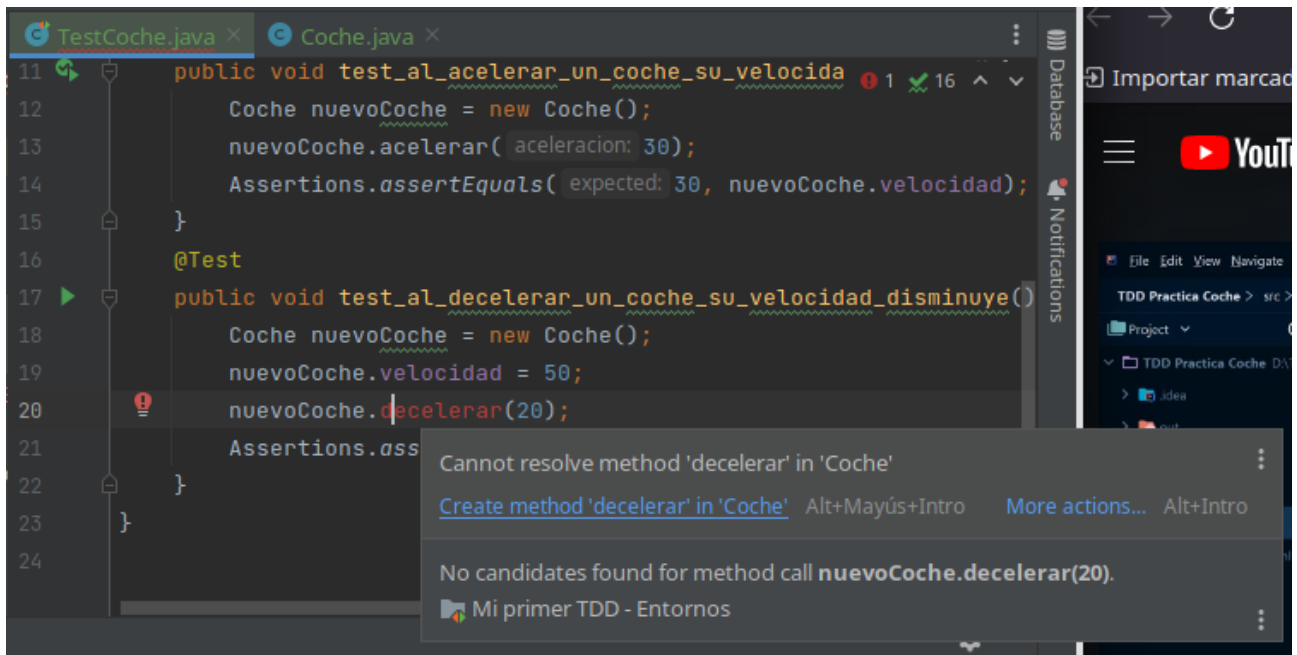


```
1 public class Coche {
2     public int velocidad;
3
4     public void acelerar(int aceleracion) {
5         velocidad+= aceleracion;
6     }
7 }
8
```

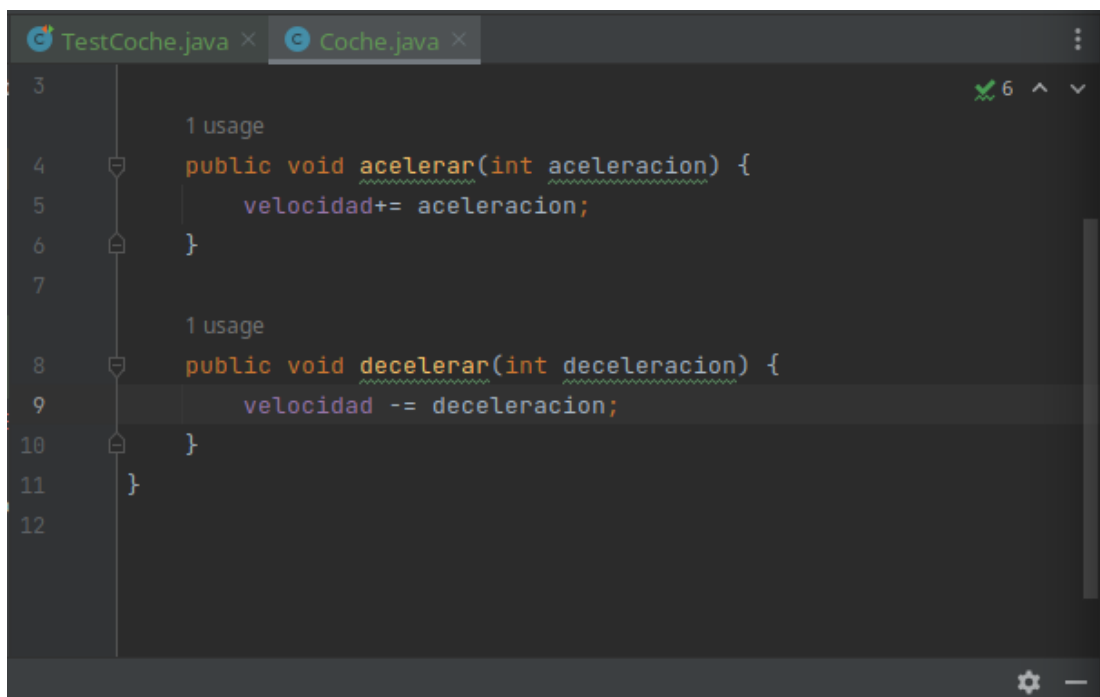
Paso 10: Ejecutamos los Test otra vez, ya que debemos comprobar que todo sigue funcionando correctamente. Ninguno de los dos Test da problemas.



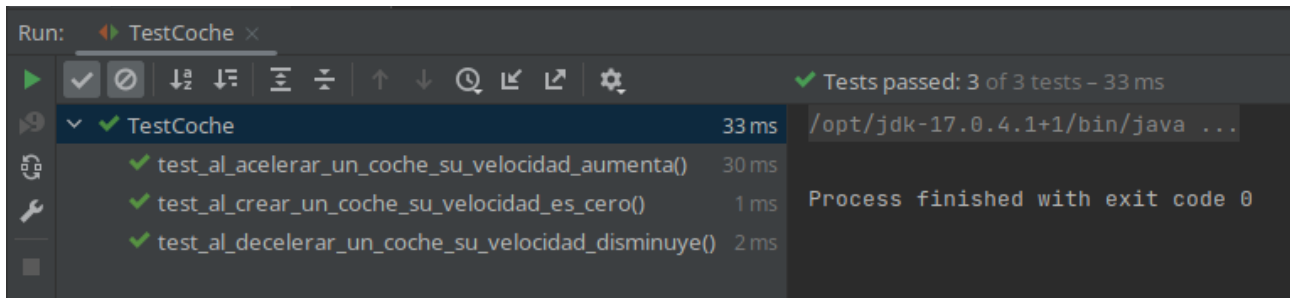
Paso 11: Creamos otro Test, esta vez se llamará "test_al_decelerar_un_coche_su_velocidad_disminuye". En este Test también se creará un objeto Coche, tendrá una velocidad con un valor de 50 y utilizará un método llamado "decelerar", con un valor de 20 pasado por parámetro. Además usa "assertEquals" igual que en el test anterior, comprueba que la velocidad de "nuevoCoche" es 30. De nuevo IntelliJ indica que "decelerar" no existe así que pulsamos para crearlo.



Vamos a la clase "Coche" y vemos que el método se ha creado, cambiamos el parámetro a "int deceleración". Dentro de este método haremos que el valor de deceleración disminuya al valor de velocidad.



Paso 12: Ejecutamos los Test y vemos que no hay ningún problema.



Paso 13: Creamos otro test que se llama

"test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero", este test es casi igual que el anterior, cambia el parámetro que le pasamos al método "decelerar", el cual es 80. También cambia el valor de velocidad esperado en "assertEquals", el cual es 0. La utilidad de este test es, como su nombre indica, comprobar que el valor velocidad no baja de 0 aunque deceleremos más de la cantidad de velocidad que "nuevoCoche" tiene.

```
TestCoche.java x Coche.java x
16 @Test
17 public void test_al_decelerar_un_coche_su_velocidad_disminuye(){
18     Coche nuevoCoche = new Coche();
19     nuevoCoche.velocidad = 50;
20     nuevoCoche.decelerar( deceleracion: 20);
21     Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
22 }
23 @Test
24 public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
25     Coche nuevoCoche = new Coche();
26     nuevoCoche.velocidad = 50;
27     nuevoCoche.decelerar( deceleracion: 80);
28     Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
29 }
30 }
```

Si ejecutamos ahora los Test, podemos ver que el que acabamos de crear da un fallo, y es debido a que el valor de velocidad es -30, y se espera que sea 0.

```
Run: TestCoche x
Tests failed: 1, passed: 3 of 4 tests - 41 ms
TestCoche 41 ms /opt/jdk-17.0.4.1+1/bin/java ...
test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero() 36 ms
test_al_acelerar_un_coche_su_velocidad_aumenta() 1 ms
test_al_crear_un_coche_su_velocidad_es_cero() 2 ms
test_al_decelerar_un_coche_su_velocidad_disminuye() 2 ms
org.opentest4j.AssertionFailedError:
Expected :0
Actual   :-30
<Click to see difference>
```

Para solucionar lo indicado antes, vamos al método "decelerar" en la clase "Coche" y dentro añadimos un "if" en el que comprobaremos si después de que se reste la deceleración a la velocidad, el valor velocidad es menor a 0, y si es el caso, velocidad se igualará a 0.

```
TestCoche.java x Coche.java x
4 public void acelerar(int aceleracion) {
5     velocidad+= aceleracion;
6 }
7
8 public void decelerar(int deceleracion) {
9     velocidad -= deceleracion;
10    if (velocidad <0) velocidad = 0;
11 }
12 }
```

Paso 14: Como último paso, ejecutamos de nuevo todos los Test para comprobar si hay algún fallo. Podemos ver que todo funciona correctamente.

