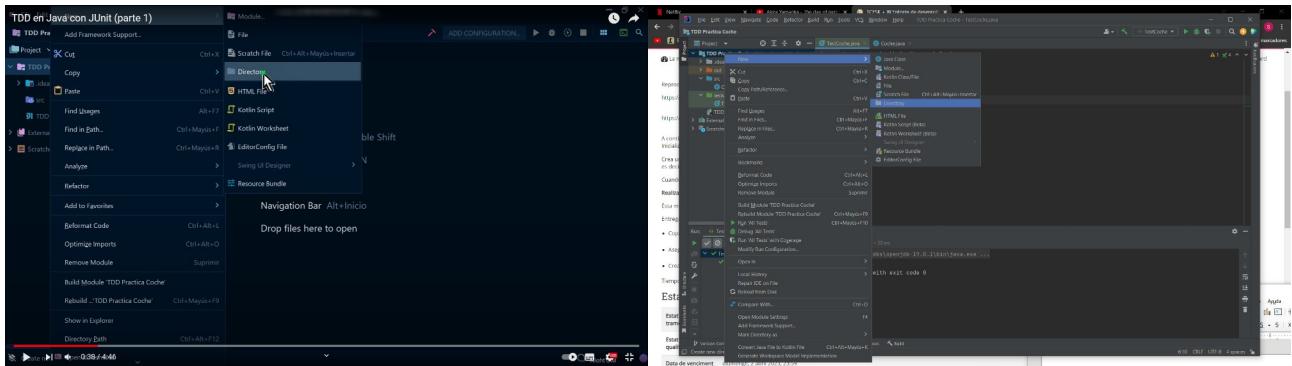


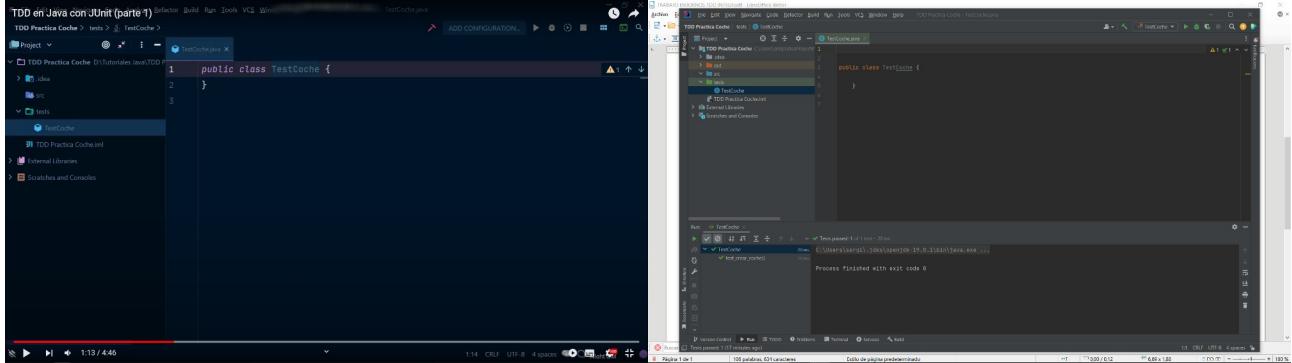
Mi primer TDD

En primer lugar crearemos el proyecto TDD Práctica Coche, a continuación crearemos un directorio y lo convertiremos en un directorio de tipo test.

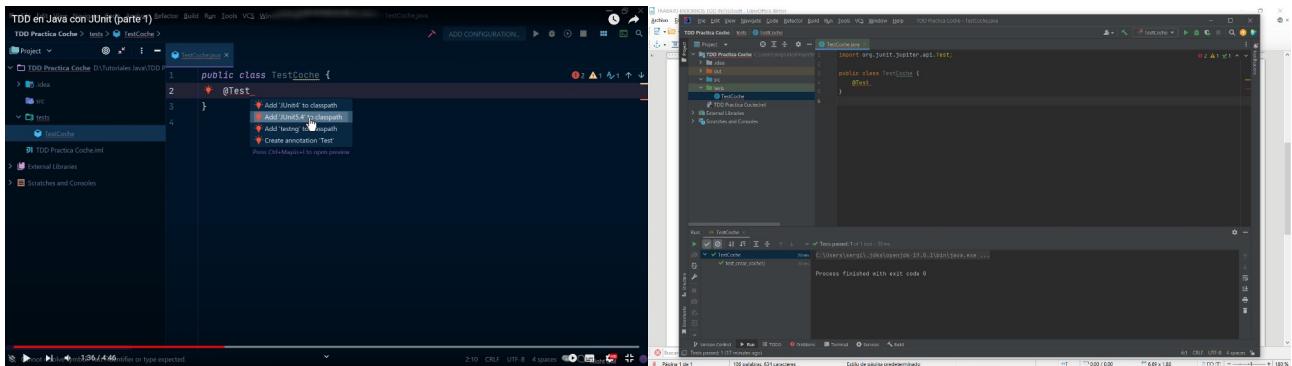


Para marcar el directorio como test hacemos click derecho sobre el y seleccionamos Mark directory as Test Sources Root y le asignamos el nombre de test en este caso.

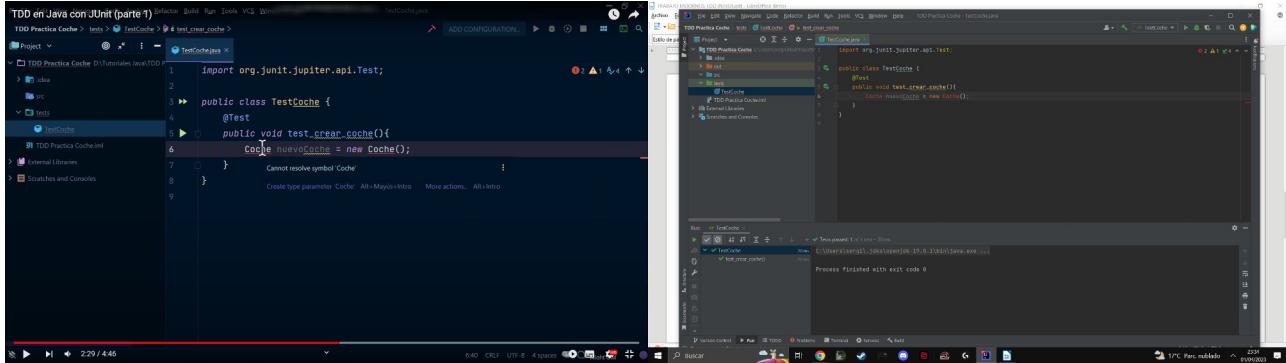
Como estamos haciendo TDD primero vamos a crear los test y acto seguido implementaremos las clases, dicho esto creamos la clase java TestCoche dentro de la carpeta tests.



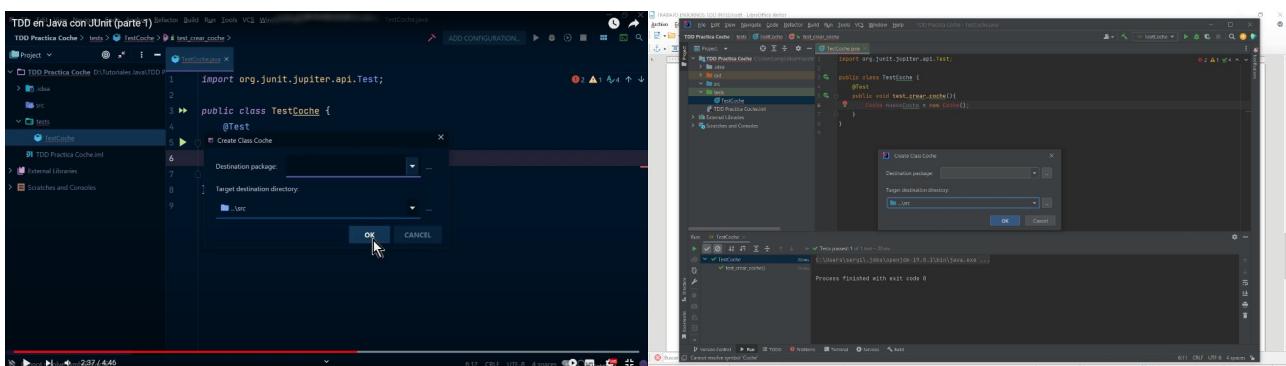
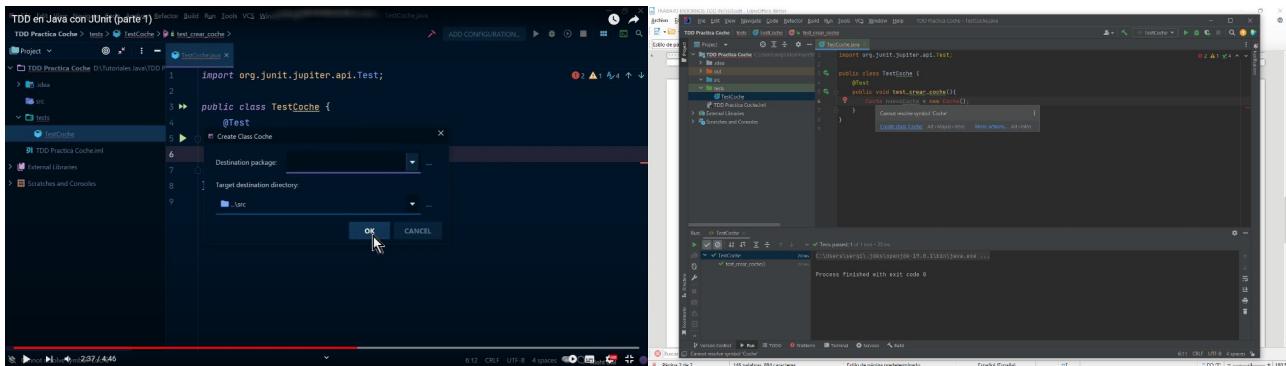
Una vez creada la clase TestCoche indicaremos que es un test mediante el `@Test` y acto seguido indicaremos que utilice Junit5 para que reconozca el `@Test`.



A continuación dentro de TestCoche crearemos test_create_coche y crearemos una clase Coche que como vemos todavía no existe.



Como Coche no existe le indicaremos que queremos crear la clase coche, yo en este caso utilizaré las facilidades de intelliJ en lugar de picar el código y seleccionare create class Coche.



Una vez con nuestra clase Coche creada y con todo el código correcto procedemos a ejecutar nuestro primer test pinchando sobre el icono de inicializar en la línea de Test Coche, como vemos se ha ejecutado sin problemas y hemos pasado el primer test por lo que procederé a realizar un control de versiones.

```

import org.junit.jupiter.api.Test;
public class TestCache {
    @Test
    public void test_crear_cache(){
        Cache nuevaCache = new Cache();
    }
}

```

Terminal output:

```

Tests passed: 1 of 1 test - 23 ms
java -version
java version "13.0.2" 2020-04-29 LTS
Java(TM) SE Runtime Environment (build 13.0.2+8-LTS)
Java HotSpot(TM) 64-Bit Server VM (build 13.0.2+8-LTS, mixed mode, sharing)
Process finished with exit code 0

```

A continuación creare un nuevo repositorio en la página de github.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner * **Repository name ***

sergikarx / Mi_primer_TDD ✓

Great repository names are taken. Mi_primer_TDD is available. Need inspiration? How about [probable-rotary-phone](#)?

Description (optional)

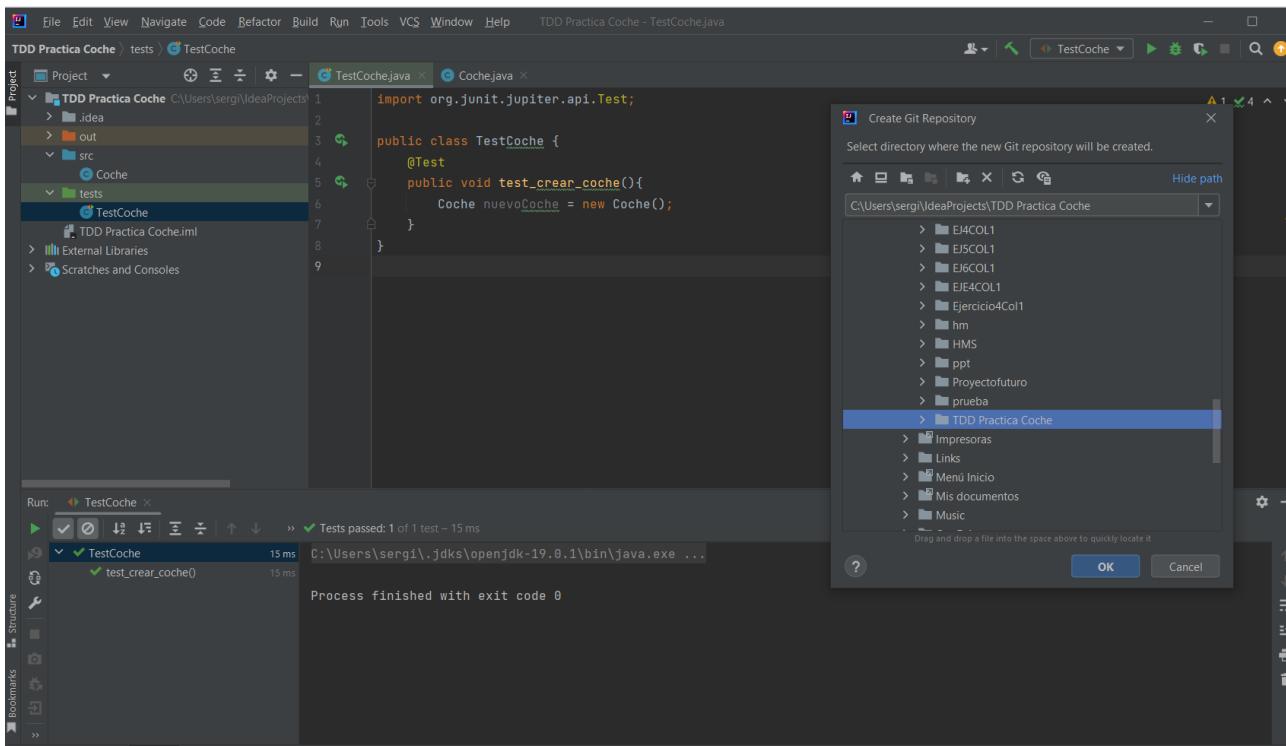
Mi primer TDD

Public
Anyone on the internet can see this repository. You choose who can commit.

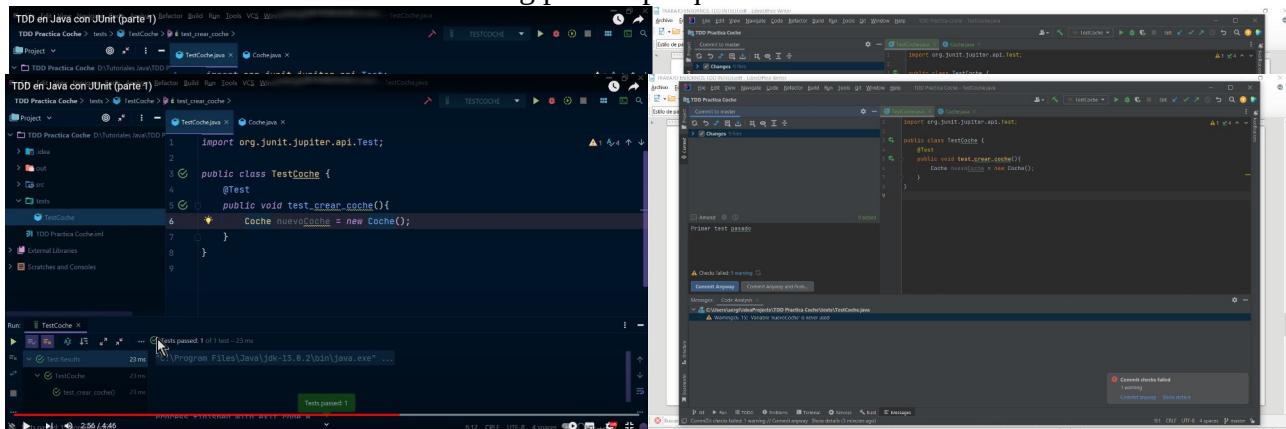
Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

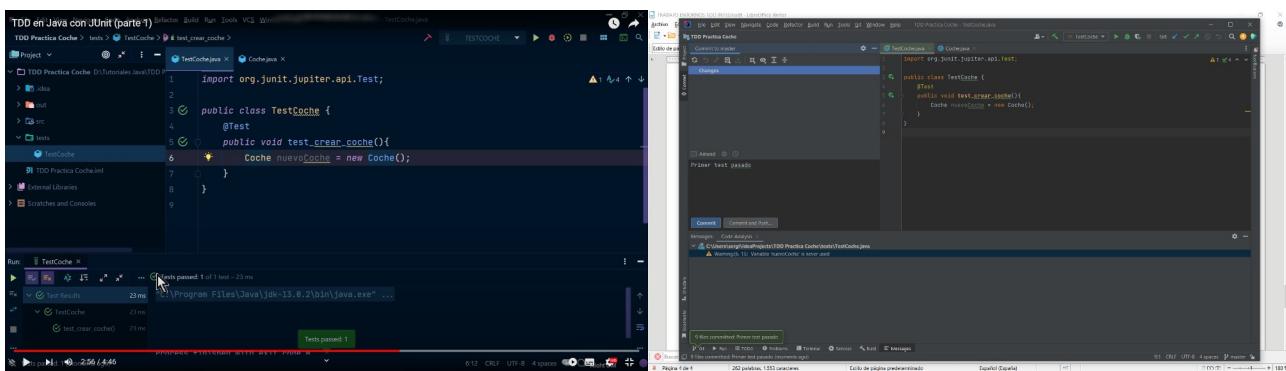
A continuación pulsaré en VCS dentro de intelij y creare un repositorio Git, seleccionaré mi proyecto, acto seguido realizaré el primer commit .



Como vemos el commit tiene un warning pero es porque la variable Coche no ha sido usada.

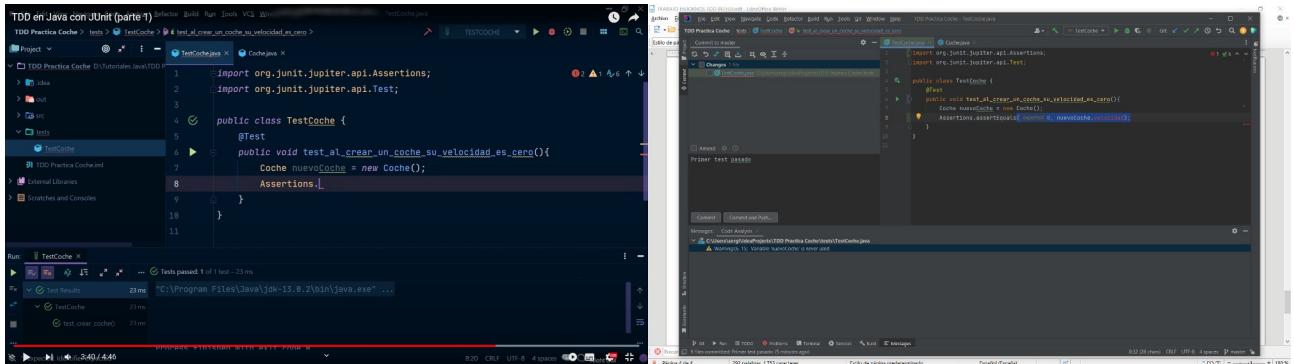


Como este warning es algo esperable pulsaremos en commit anyway y realizamos nuestro commit. Commit realizado correctamente.



A continuación vamos a mejorar el test y modificaremos el método crear coche.

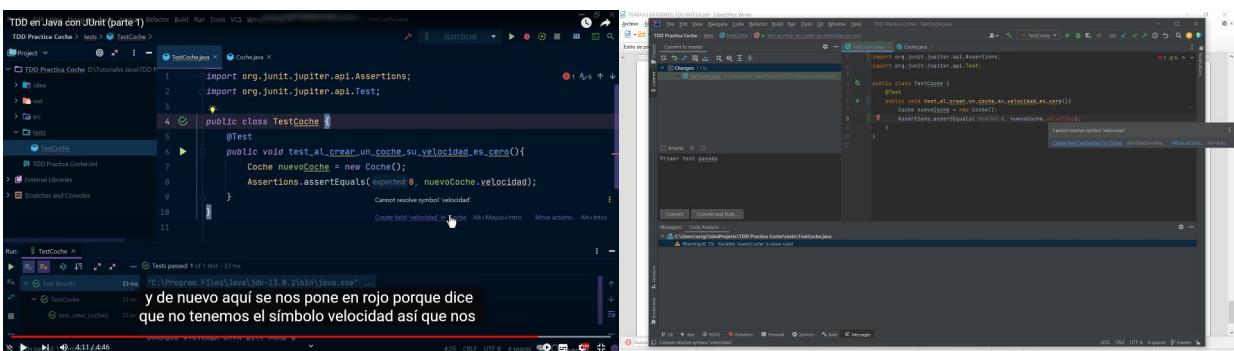
Para el nuevo método importaremos la clase Assertions porque necesitaremos una aserción y crearemos nuestro nuevo método.



```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class TestCoche {
    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }
}
```

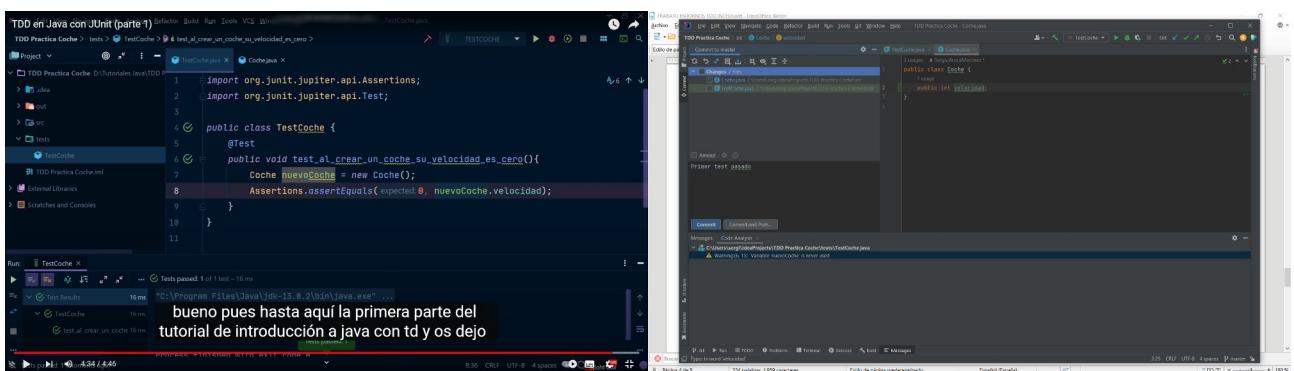
Como vemos el simbolo velocidad del nuevo método no existe por lo tanto lo crearemos.



```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class TestCoche {
    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }
}
```

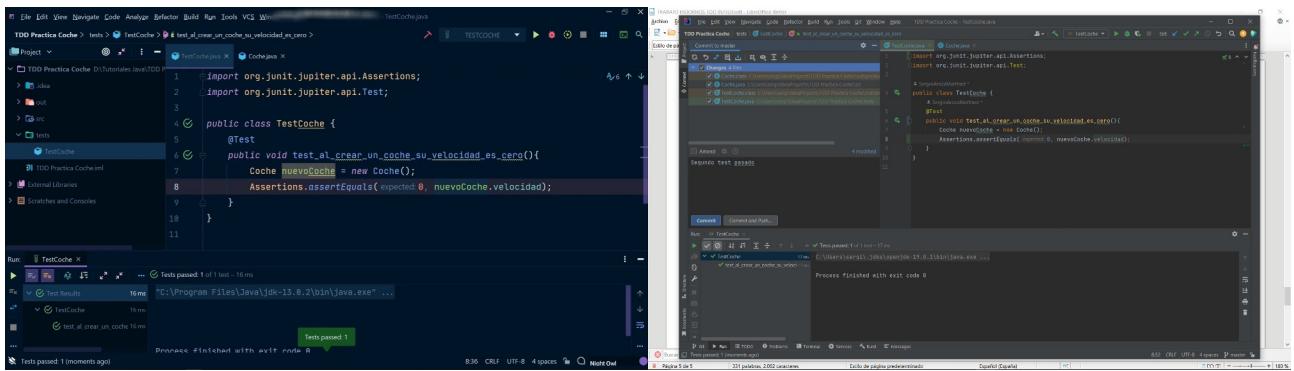
y de nuevo aquí se nos pone en rojo porque dice que no tenemos el símbolo velocidad así que nos



```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class TestCoche {
    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }
}
```

Ahora procederemos a pasar un segundo test.

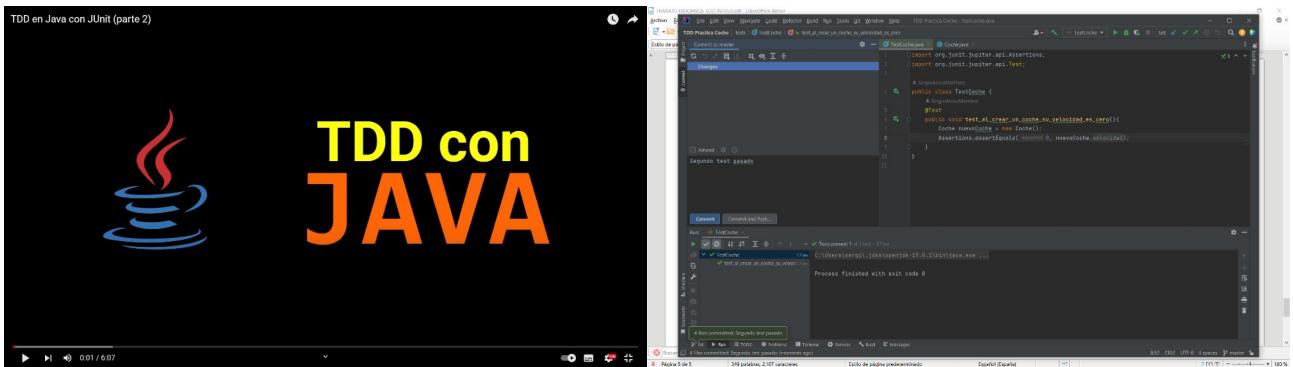


```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

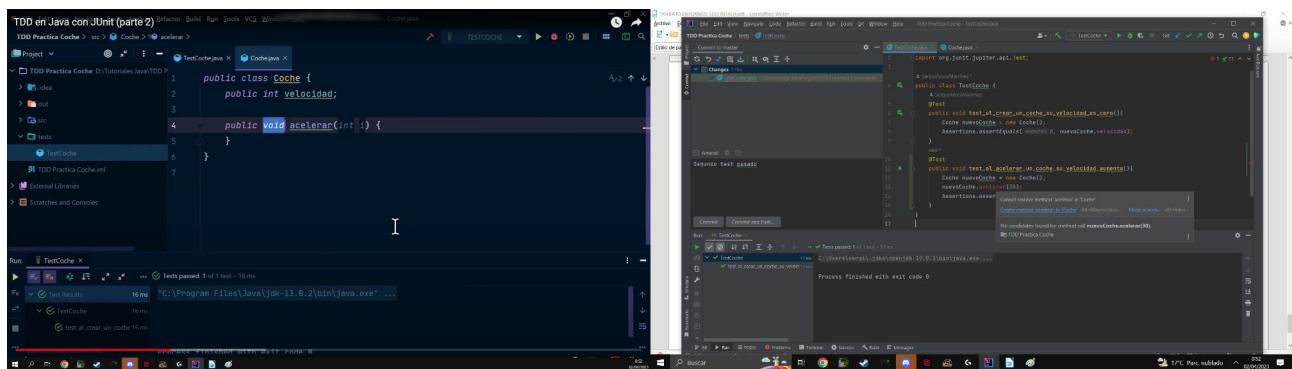
public class TestCoche {
    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }

    @Test
    public void test_al_crear_un_coche_su_velocidad_no_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertNotEquals(0, nuevoCoche.velocidad);
    }
}
```

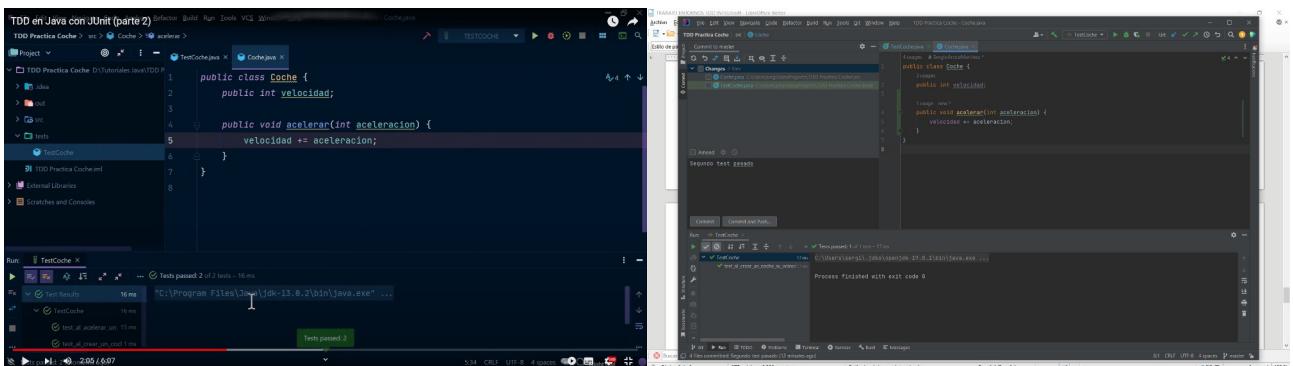
Como vemos el test se ha pasado correctamente por lo que ahora procederemos a realizar un segundo commit.



A continuación crearemos el método acelerar para cambiar la velocidad del coche.

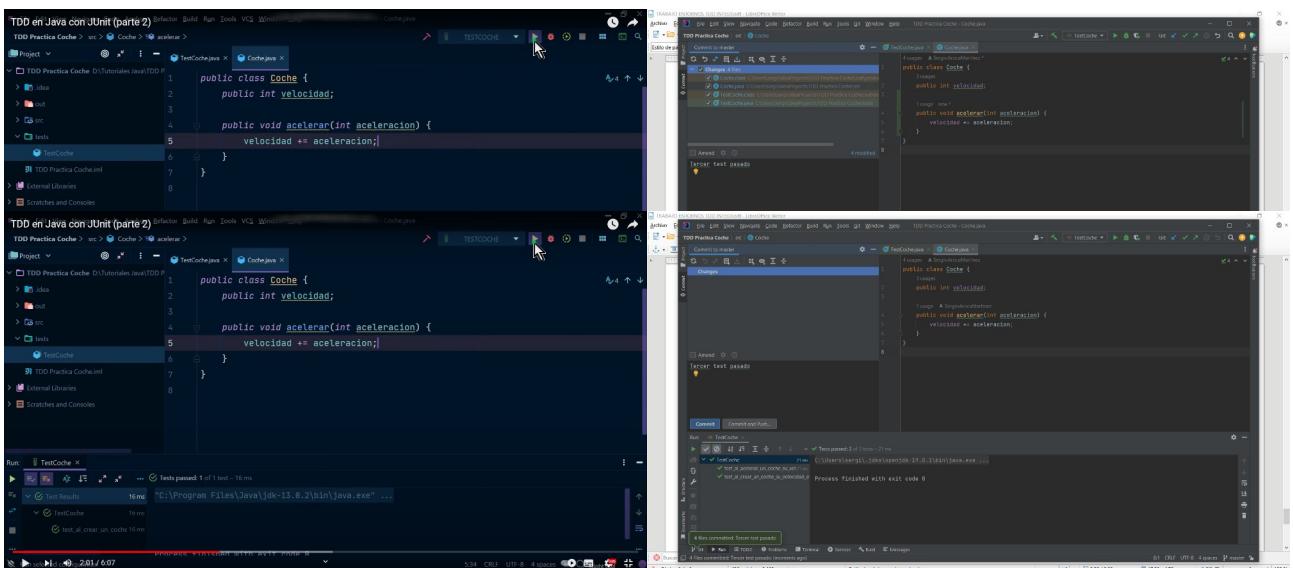


Como vemos acelerar no existe por lo que lo crearemos como hemos hecho anteriormente con velocidad.



A continuación pasaremos otro test y realizaremos un tercer commit.

Como vemos ha pasado el de nuevo el test correctamente por lo que ahora es momento de realizar el tercer commit.



A continuación crearemos un tercer método llamado decelerar

The screenshot shows an IDE interface with two tabs open: `Coche.java` and `TestCoche.java`. In `Coche.java`, a new method `decelerar` is being added:public class Coche { public int velocidad; public void acelerar(int aceleracion) { velocidad += aceleracion; } public void decelerar(int deceleracion) { velocidad -= deceleracion; }}In `TestCoche.java`, a new test `test_decelerar` is being written:public class TestCoche { @Test public void test_decelerar() { Coche coche = new Coche(); coche.acelerar(10); coche.decelerar(5); Assertions.assertEquals(5, coche.velocidad); }}The status bar at the bottom indicates "Tests passed: 2 (in 0 minutes ago)".

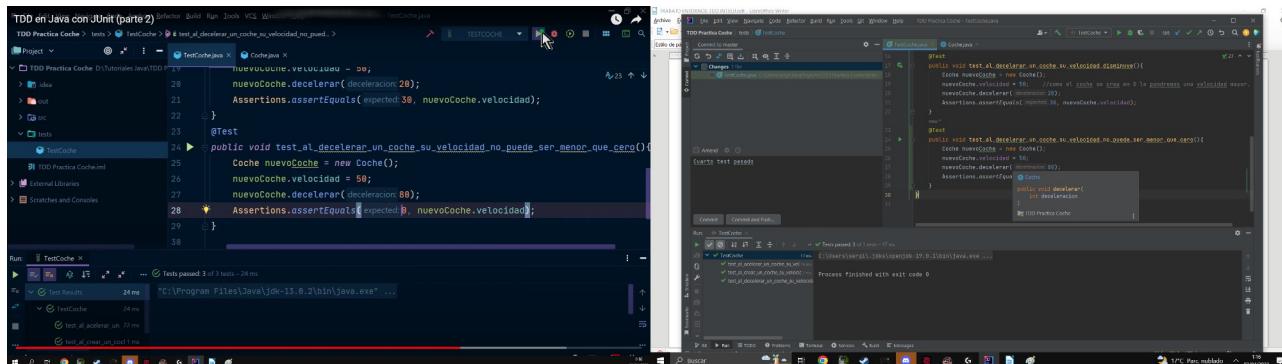
Al igual que anteriormente nos falta decelerar en la clase Coche, por lo que habrá que crearlo como anteriormente hemos hecho.

The screenshot shows the IDE after the `decelerar` method has been implemented in `Coche.java`:public class Coche { public int velocidad; public void acelerar(int aceleracion) { velocidad += aceleracion; } public void decelerar(int deceleracion) { velocidad -= deceleracion; }}The `TestCoche.java` file now contains three tests, all of which have passed:public class TestCoche { @Test public void test_acelerar() { Coche coche = new Coche(); coche.acelerar(10); Assertions.assertEquals(10, coche.velocidad); } @Test public void test_decelerar() { Coche coche = new Coche(); coche.acelerar(10); coche.decelerar(5); Assertions.assertEquals(5, coche.velocidad); } @Test public void test_crear_un_coche() { Coche coche = new Coche(); Assertions.assertNotNull(coche); }}The status bar at the bottom indicates "Tests passed: 3 / 3 ms (in 0 minutes ago)".

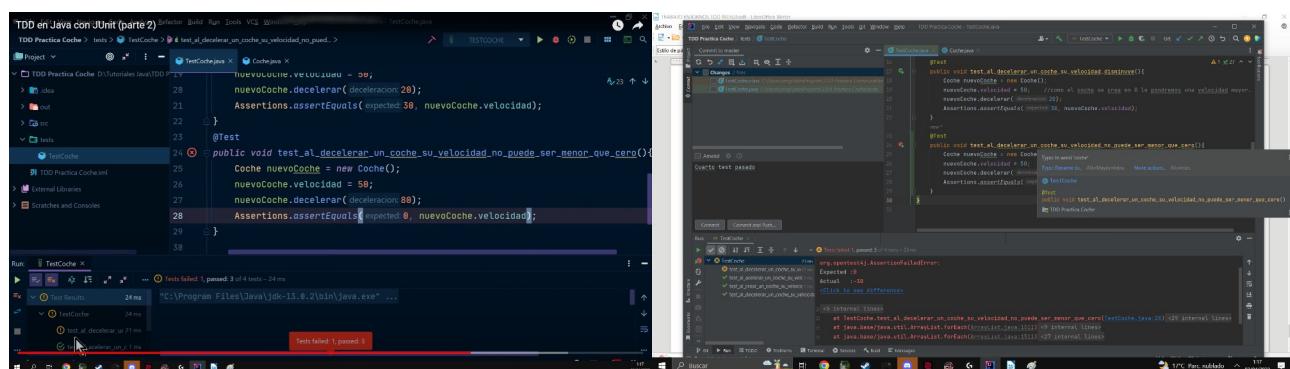
Como vemos se ha pasado el test correctamente por lo que es momento de realizar de nuevo un commit.

The screenshot shows the IDE after the `decelerar` method has been implemented in `Coche.java`:public class Coche { public int velocidad; public void acelerar(int aceleracion) { velocidad += aceleracion; } public void decelerar(int deceleracion) { velocidad -= deceleracion; }}The `TestCoche.java` file now contains three tests, all of which have passed:public class TestCoche { @Test public void test_acelerar() { Coche coche = new Coche(); coche.acelerar(10); Assertions.assertEquals(10, coche.velocidad); } @Test public void test_decelerar() { Coche coche = new Coche(); coche.acelerar(10); coche.decelerar(5); Assertions.assertEquals(5, coche.velocidad); } @Test public void test_crear_un_coche() { Coche coche = new Coche(); Assertions.assertNotNull(coche); }}The status bar at the bottom indicates "Tests passed: 3 / 3 ms (in 0 minutes ago)".

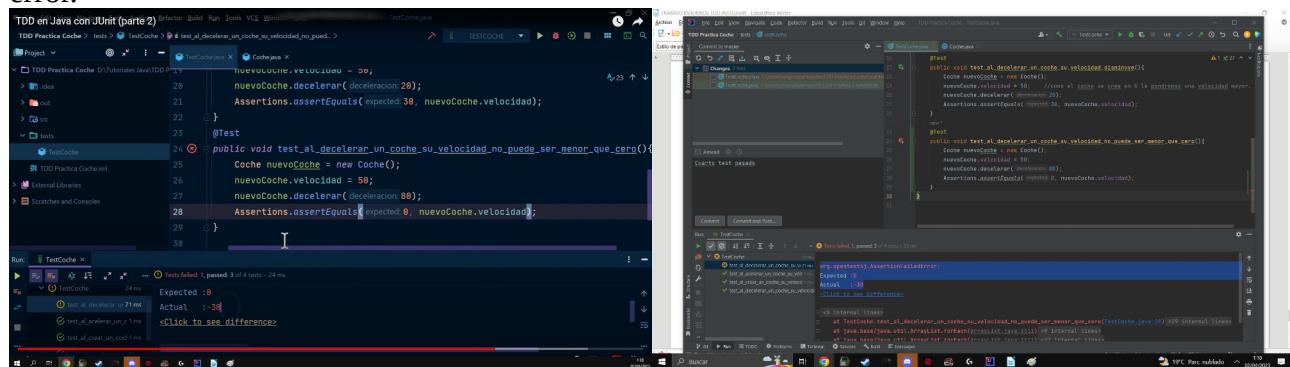
Por último crearemos un nuevo test que va a servir para que al decelerar un coche su velocidad no pueda ser menor que 0.



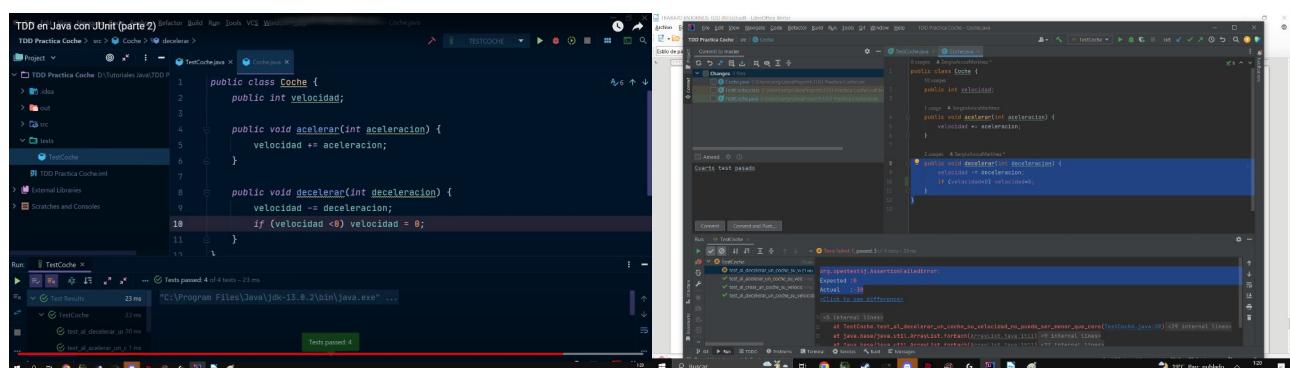
Después de crear este método intentaremos pasar el test pero a pesar de poderse compilar nos dará error:



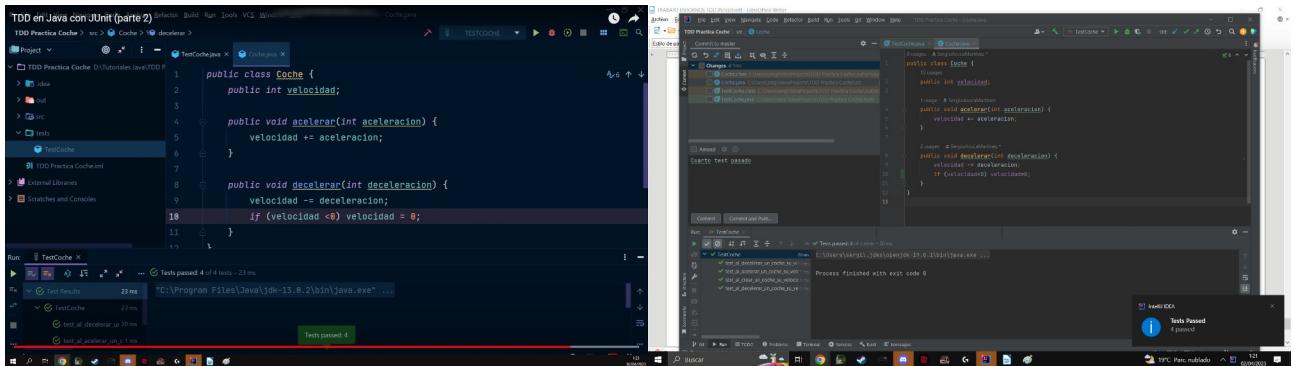
Como vemos en el historial de test sale el test fallido y si clickeamos en el nos dará más detalles del error.



Como vemos la velocidad es menor que 0 por lo que faremos unas modificaciones en el código añadiendo un condicional en decelerar para que la velocidad no pueda ser menor que 0.

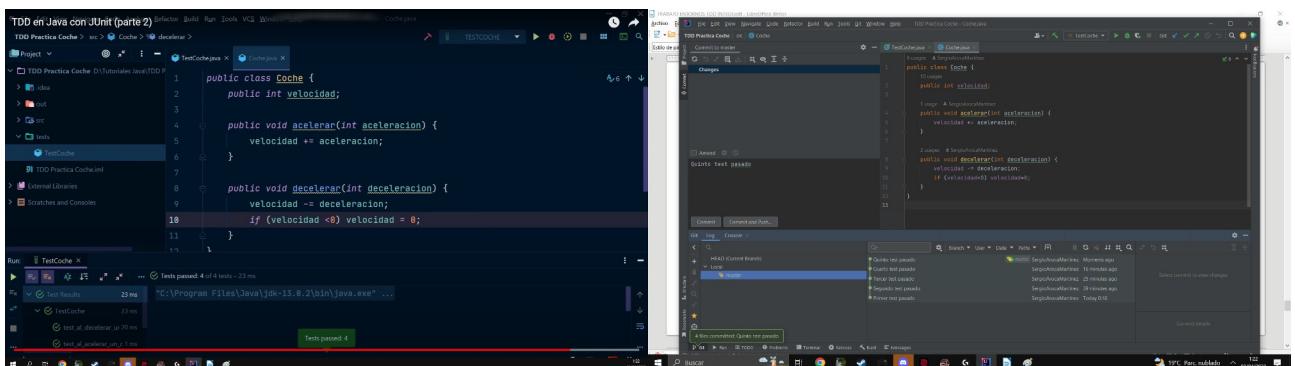


Como vemos hemos pasado correctamente el test por lo que procederemos a realizar commit como anteriormente.



```
public class Coche {
    public int velocidad;
    public void acelerar(int aceleracion) {
        velocidad += aceleracion;
    }
    public void decelerar(int deceleracion) {
        velocidad -= deceleracion;
        if (velocidad <= 0) velocidad = 0;
    }
}
```

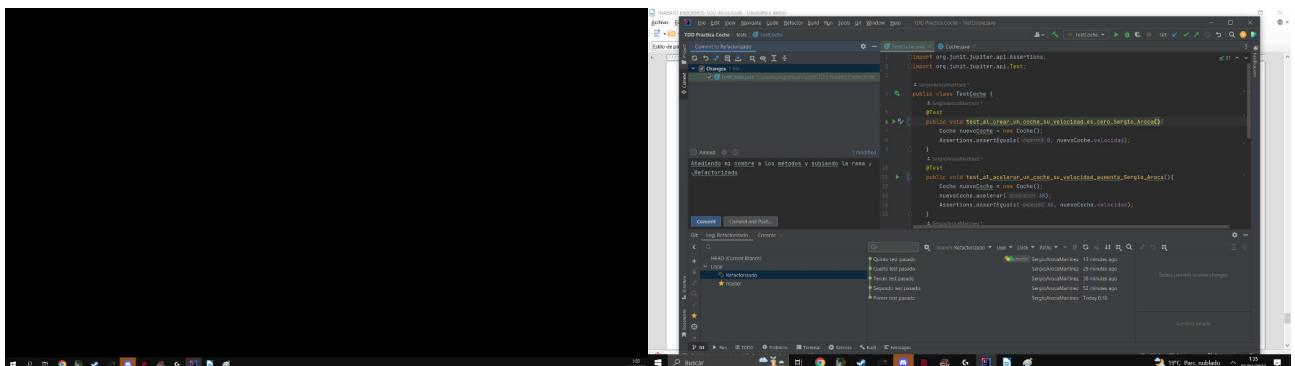
```
public class TestCoche {
    @Test
    public void test_acelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        assertEquals(10, coche.velocidad);
    }
    @Test
    public void test_decelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        coche.decelerar(10);
        assertEquals(0, coche.velocidad);
    }
}
```



```
public class Coche {
    public int velocidad;
    public void acelerar(int aceleracion) {
        velocidad += aceleracion;
    }
    public void decelerar(int deceleracion) {
        velocidad -= deceleracion;
        if (velocidad <= 0) velocidad = 0;
    }
}
```

```
public class TestCoche {
    @Test
    public void test_acelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        assertEquals(10, coche.velocidad);
    }
    @Test
    public void test_decelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        coche.decelerar(10);
        assertEquals(0, coche.velocidad);
    }
}
```

A continuación como indica el ejercicio añadiremos el nombre a los Test método creados y subiremos las dos ramas.



```
public class Coche {
    public int velocidad;
    public void acelerar(int aceleracion) {
        velocidad += aceleracion;
    }
    public void decelerar(int deceleracion) {
        velocidad -= deceleracion;
        if (velocidad <= 0) velocidad = 0;
    }
}
```

```
public class TestCoche {
    @Test
    public void test_acelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        assertEquals(10, coche.velocidad);
    }
    @Test
    public void test_decelerar() {
        Coche coche = new Coche();
        coche.acelerar(10);
        coche.decelerar(10);
        assertEquals(0, coche.velocidad);
    }
}
```

