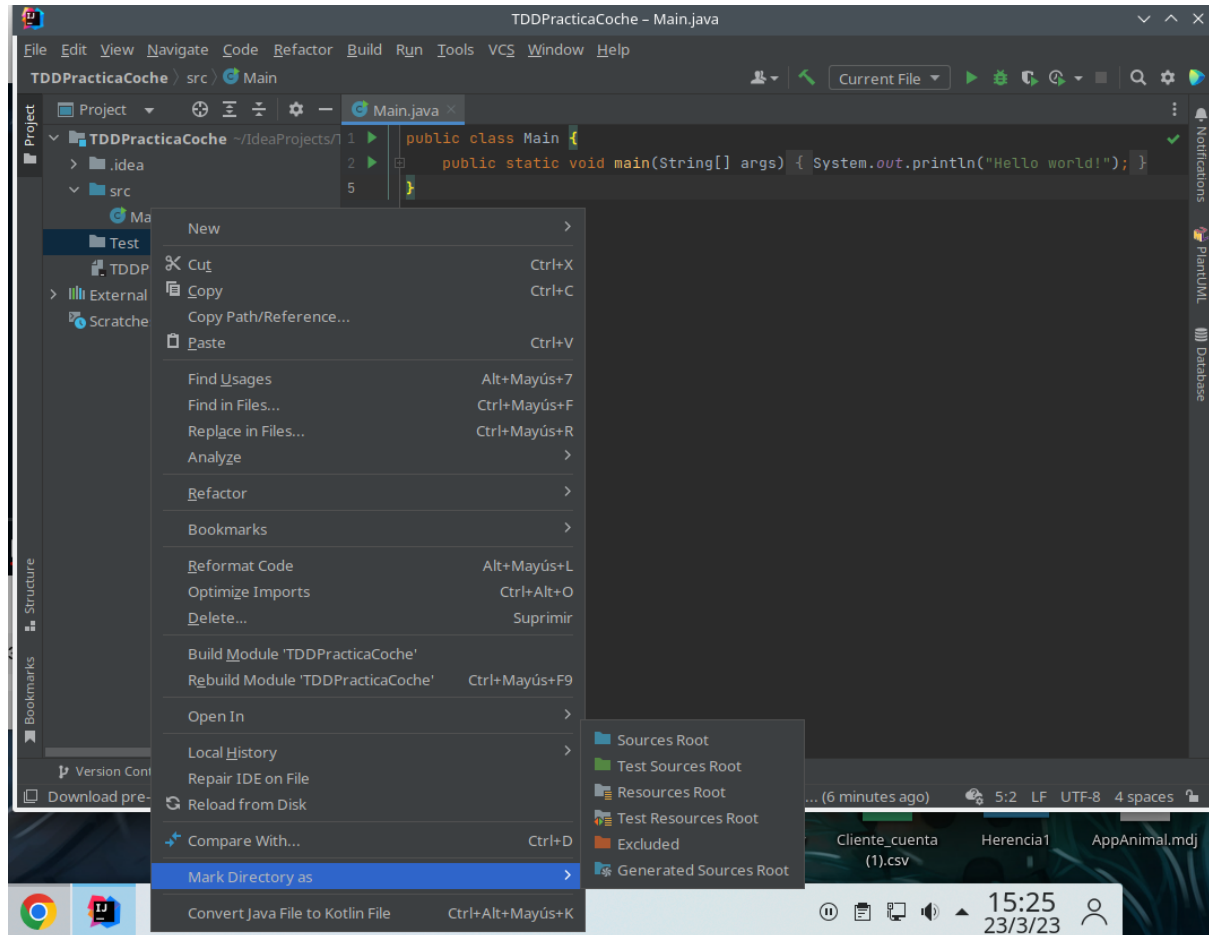
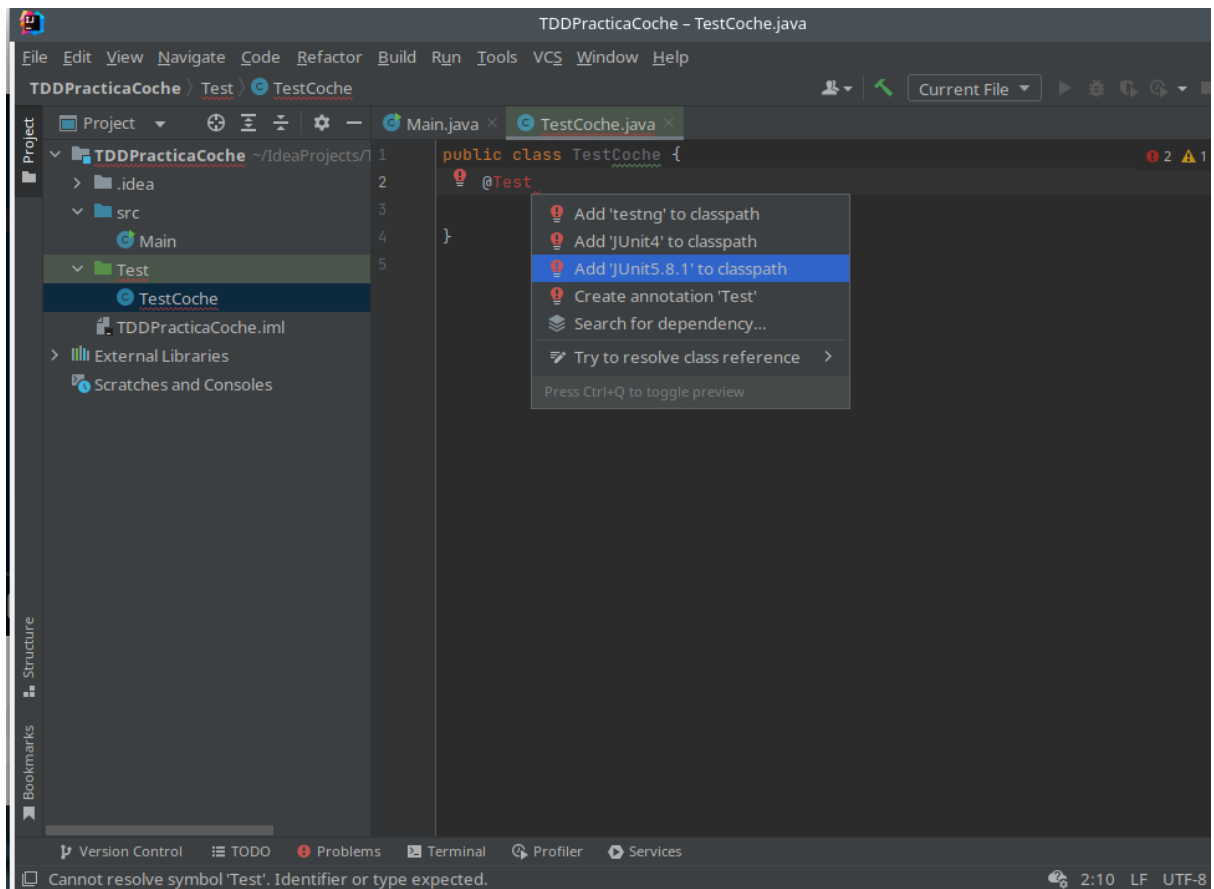


## ENTORNOS: Mi primer TDD

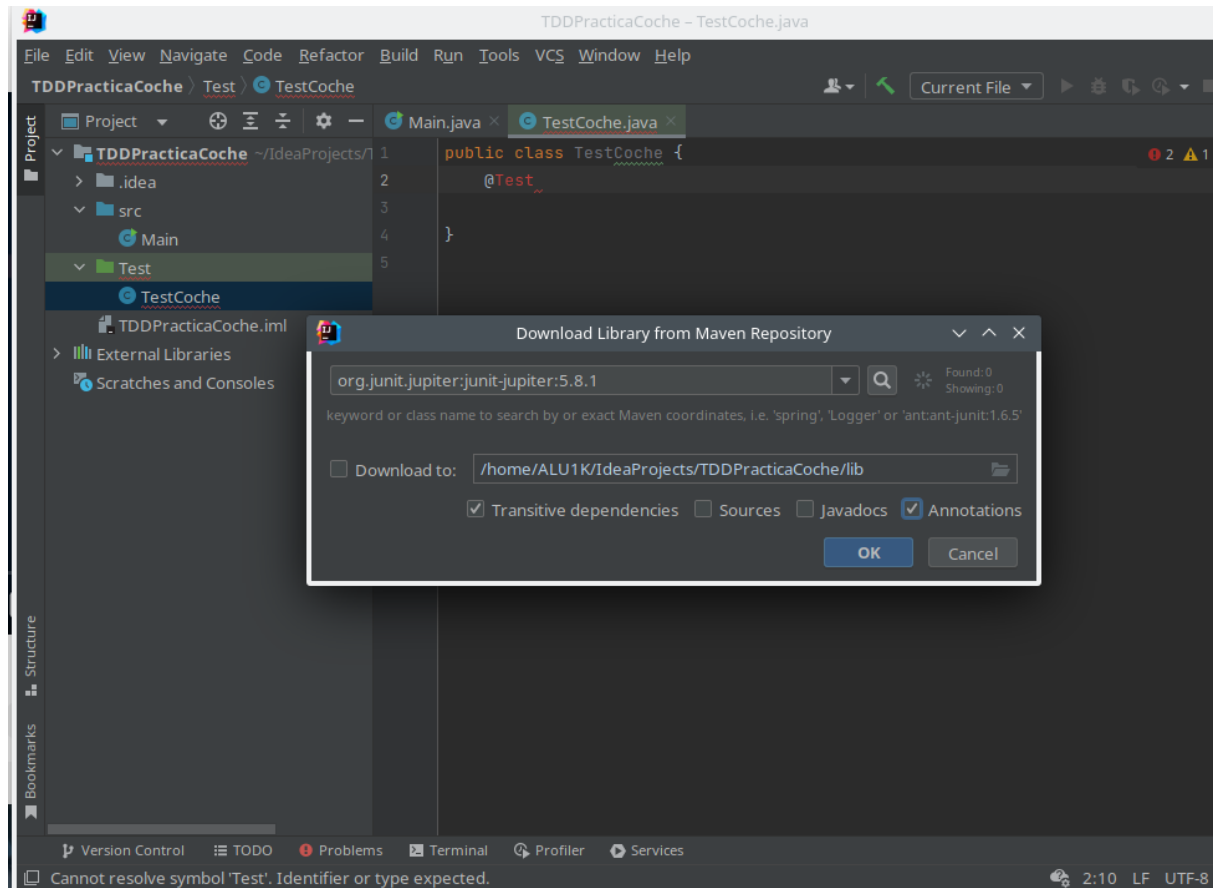
Antes de comenzar con esta práctica, crearemos un proyecto. Dentro del mismo proyecto, hacemos click derecho y creamos un directorio al que llamaremos “Test”. Una vez creado, hacemos click derecho sobre él y abajo del todo desplegamos la opción “Mark Directory as...” y elegimos “Test Sources Root”.



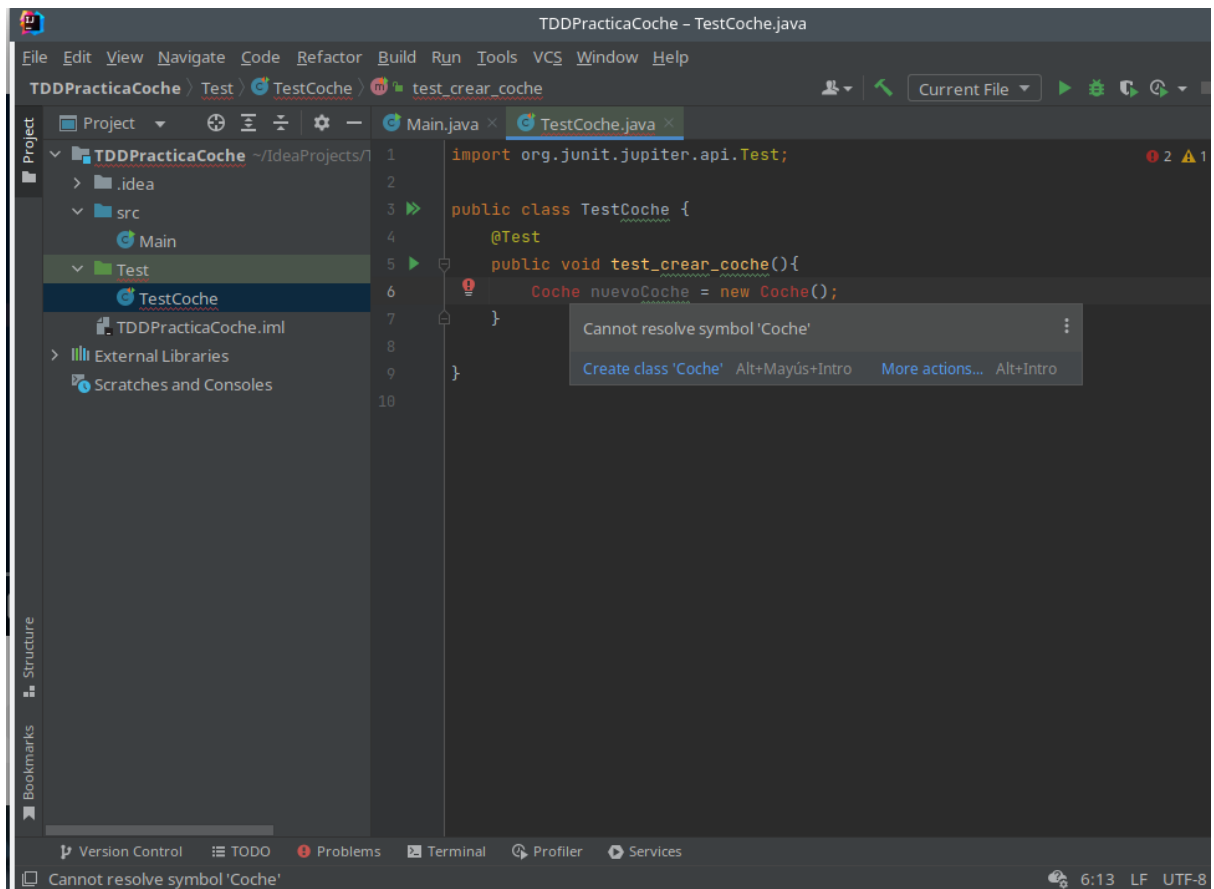
Creamos una clase dentro de Test llamada “TestCoche” y dentro de ella escribimos @Test. Al hacer alt+intro nos aparecerá este desplegable, en él elegimos la opción “Add JUnit5.8.1 to classpath” en caso de no tenerla podemos usar JUnit4.



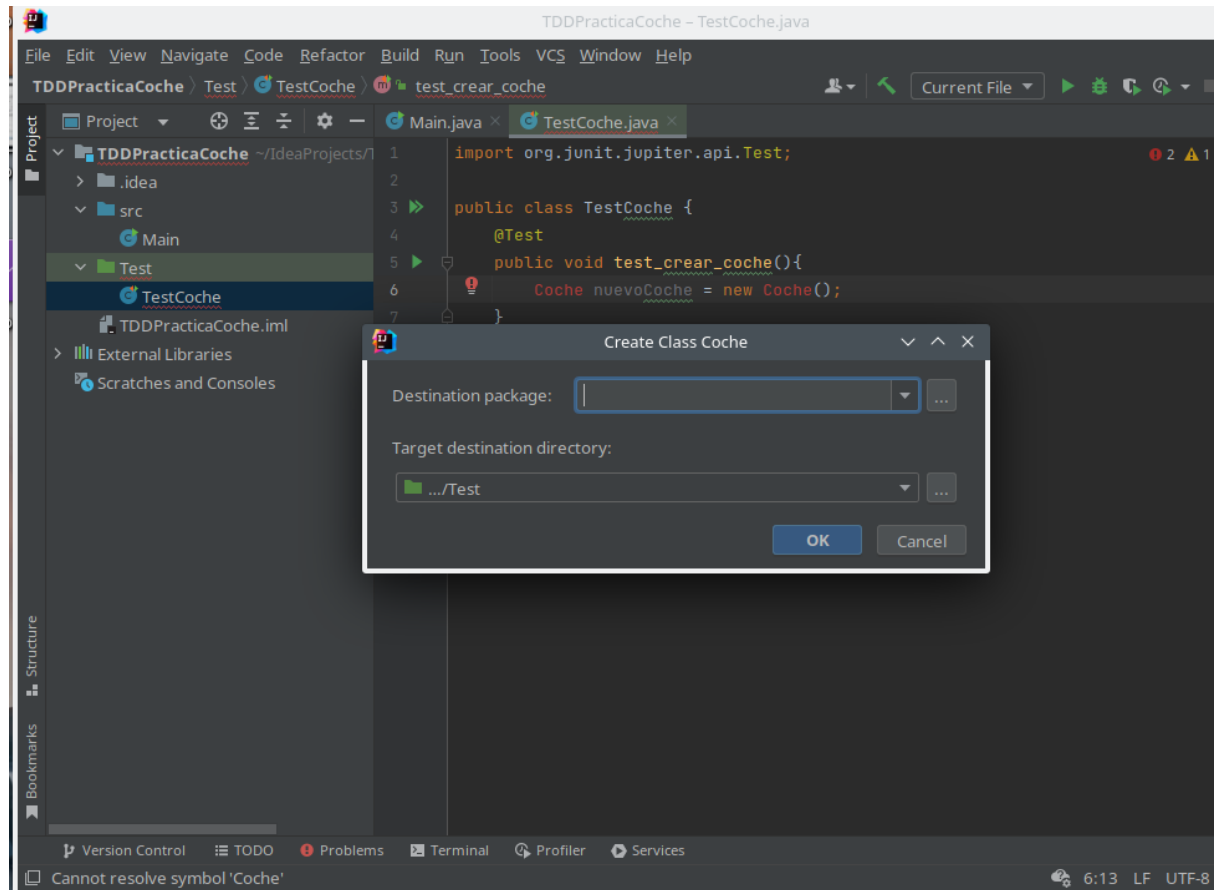
En la siguiente ventana emergente marcamos la casilla “Annotations” y aceptamos.



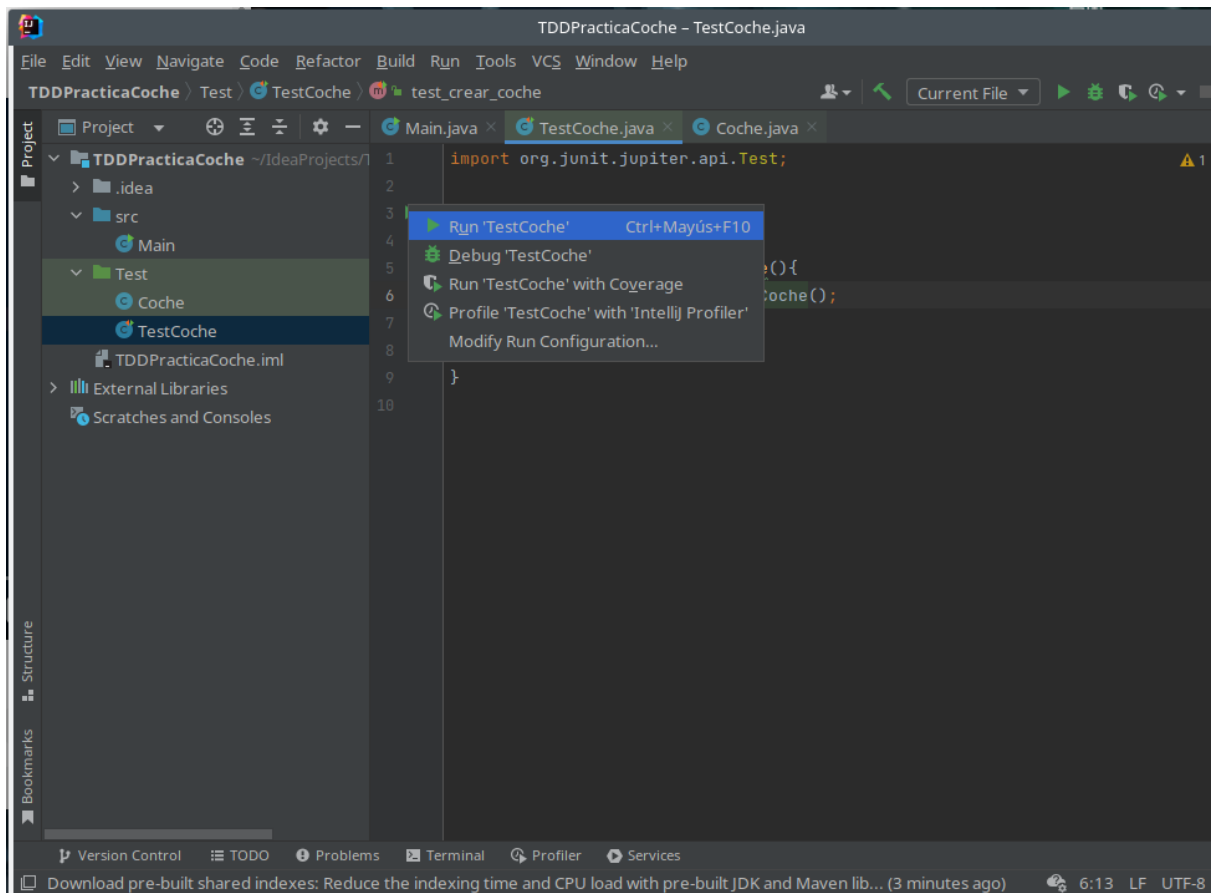
Creamos el siguiente método y en él crearemos un objeto Coche, pero este no existe por lo que aparecerá en rojo, clicamos sobre él y acercamos el ratón, nos sugerirá en una ventanita abajo a la izquierda crear la clase (Create class 'Coche'), clicamos en esa opción.



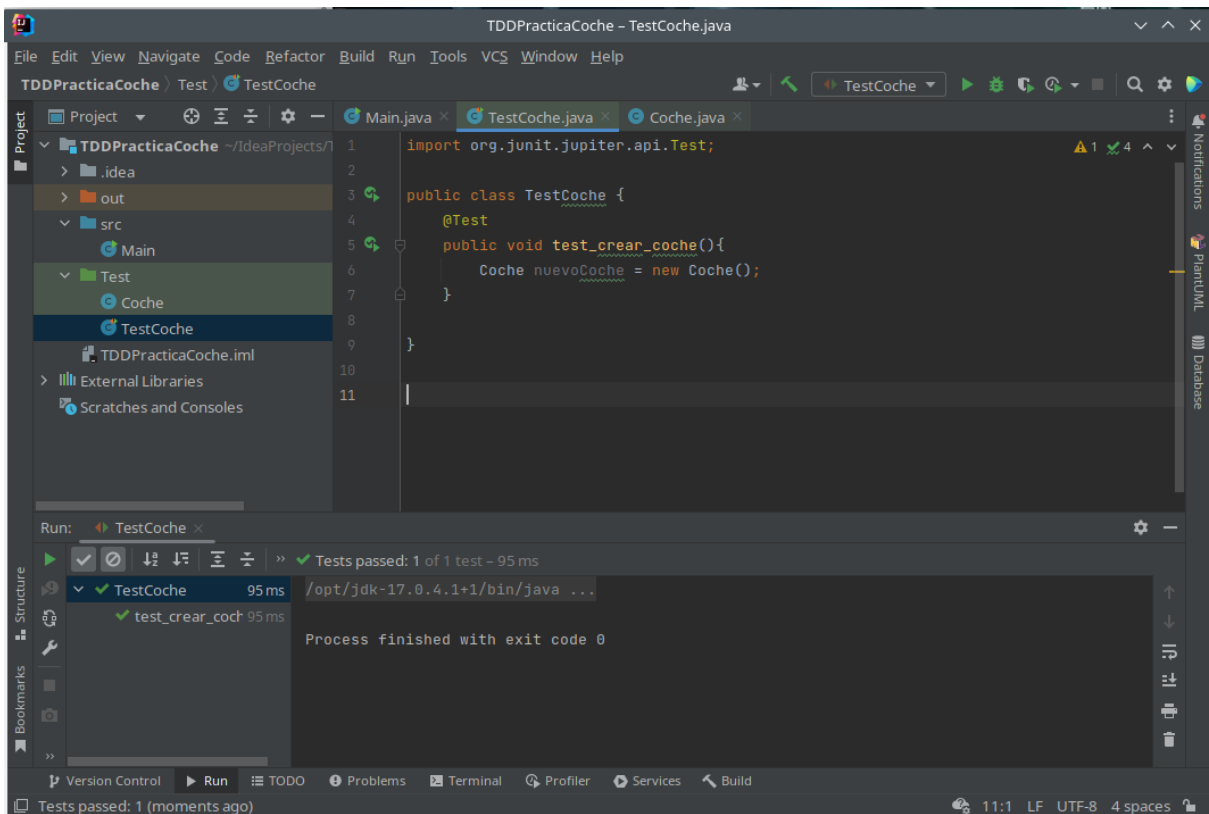
Aceptamos la ventana emergente.



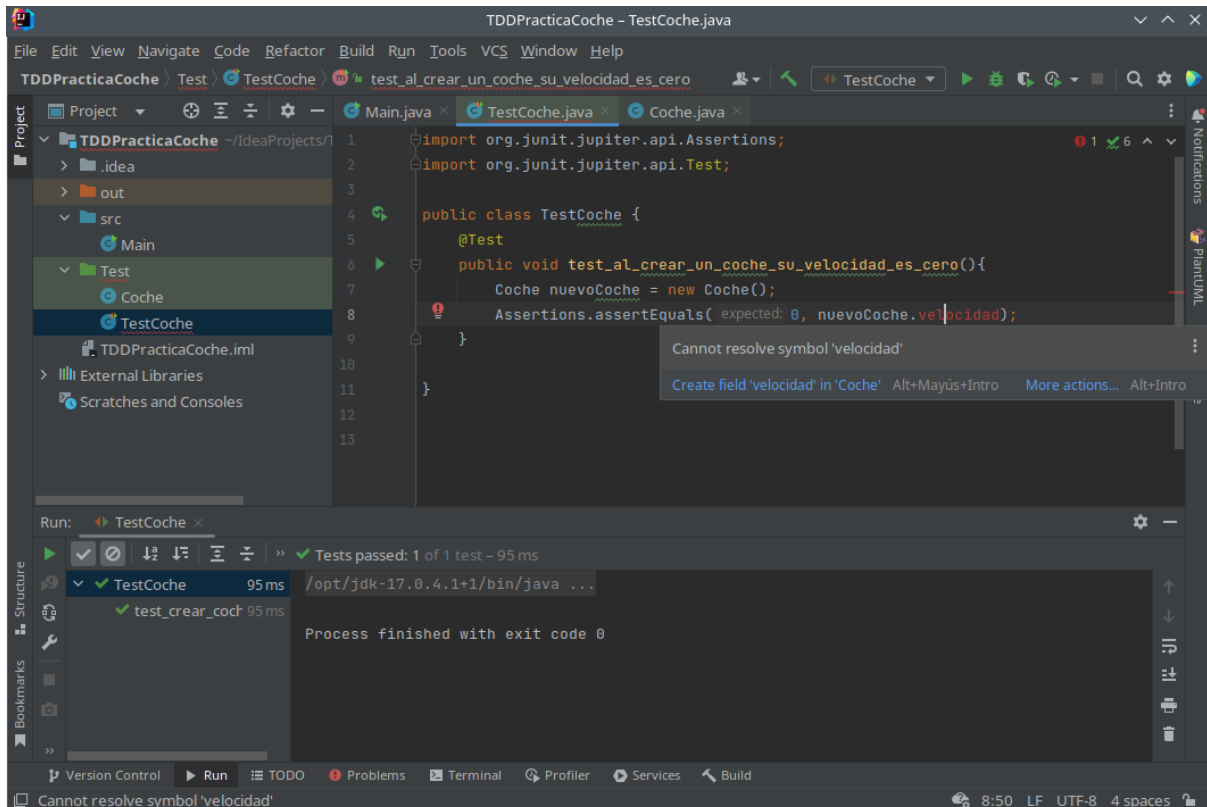
Y una vez creado el objeto clicamos sobre el botón de play verde al lado izquierdo del método y realizará la prueba de dicho método si clicamos en “Run “TestCoche”.



Si aparece en la nueva pestaña de abajo un tick verde es porque el método ha pasado la prueba.

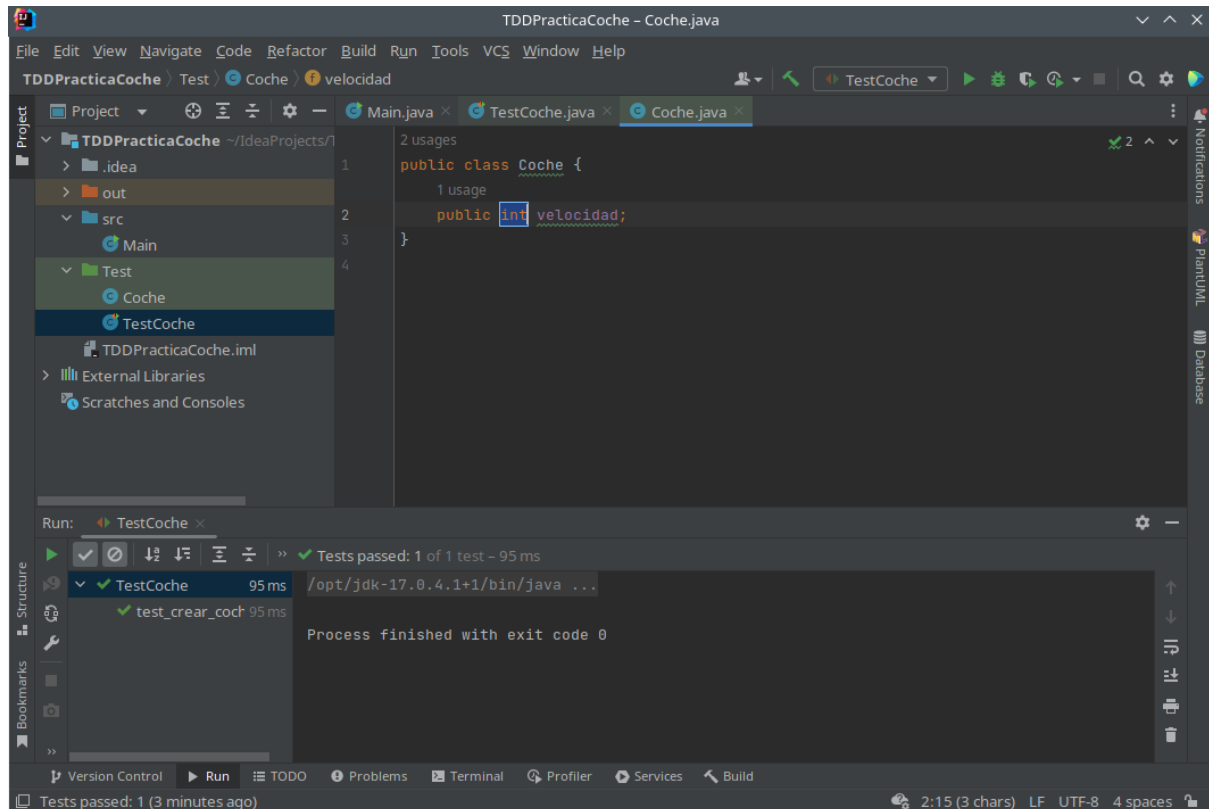


Ahora el método a parte de crear un coche lo creará con una velocidad igual a cero, por lo que escribimos `Assertions.assertEquals` que comprueba que una vez creado el coche, la velocidad de este sea igual a cero. Pero aparecerá en rojo la velocidad del coche ya que este no tiene dicho atributo, si nos colocamos en velocidad nos sugerirá crear el atributo así que lo creamos.

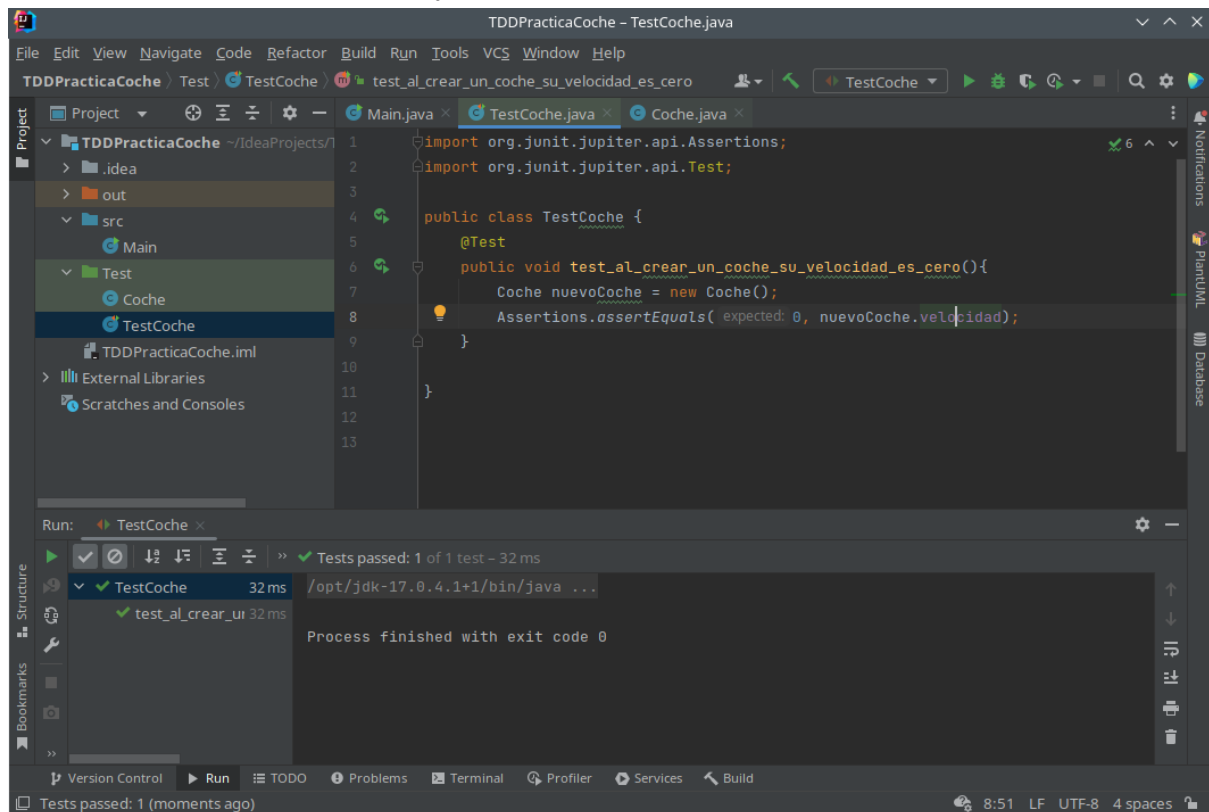




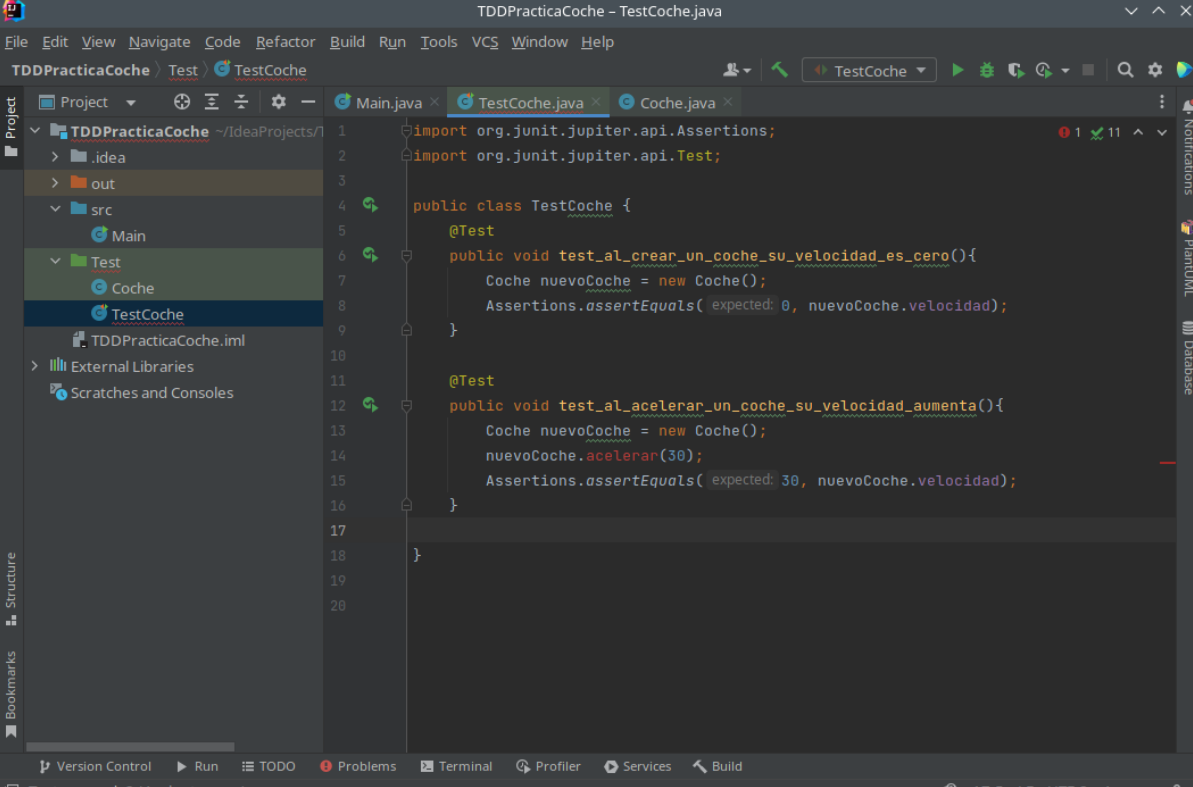
Se creará y además nos sugerirá que sea de tipo int.



Realizaremos otra vez la prueba y la aprobaremos.



Creamos un segundo método en el que crearemos un coche y llamaremos a un método de este objeto, llamado acelerar, para que aumente su velocidad dependiendo de lo que le pasemos por parámetro y por lo tanto el método que comprueba que la velocidad fuera igual a 0 ahora tiene que ser igual a lo que se haya acelerado nuestro coche, este caso tiene que ser igual a 30 porque le hemos pasado por parámetro 30.

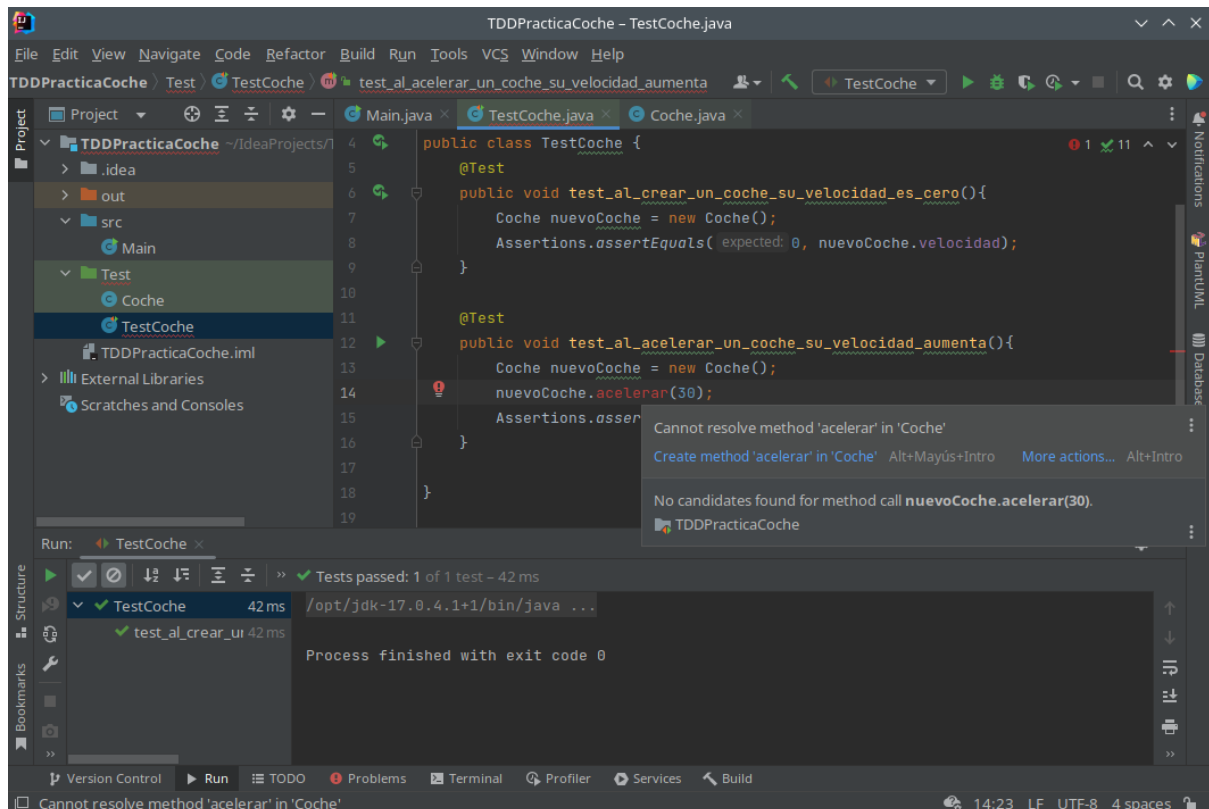


The screenshot shows an IDE window titled "TDDPracticaCoche - TestCoche.java". The left sidebar displays the project structure with "TDDPracticaCoche" as the root, containing "src" (with "Main" and "Coche") and "Test" (with "TestCoche"). The main editor shows the following Java code:

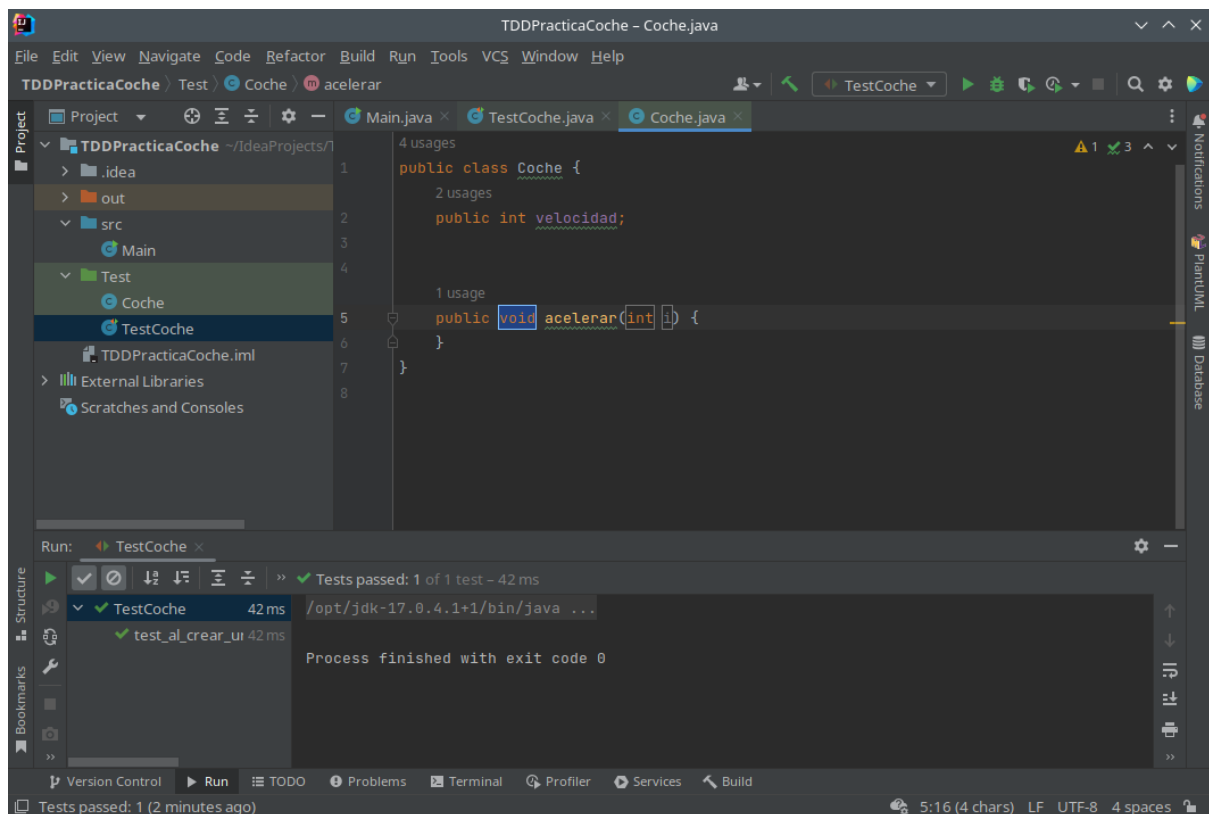
```
1 import org.junit.jupiter.api.Assertions;
2 import org.junit.jupiter.api.Test;
3
4 public class TestCoche {
5     @Test
6     public void test_al_crear_un_coche_su_velocidad_es_cero(){
7         Coche nuevoCoche = new Coche();
8         Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);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

The bottom status bar indicates "Tests passed: 2 (4 minutes ago)" and "17:5 LF UTF-8 4 spaces".

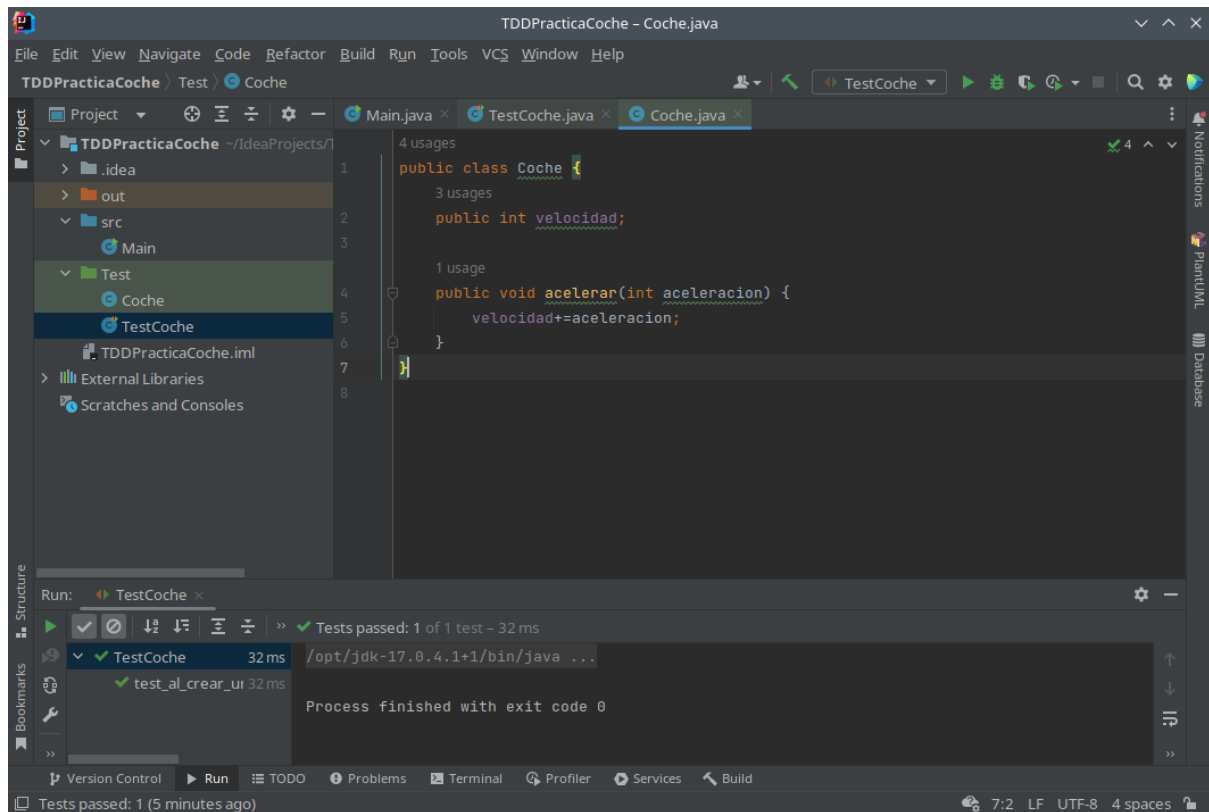
El método acelerar aparecerá en rojo porque no existe, así que como ya hemos hecho anteriormente lo creamos.



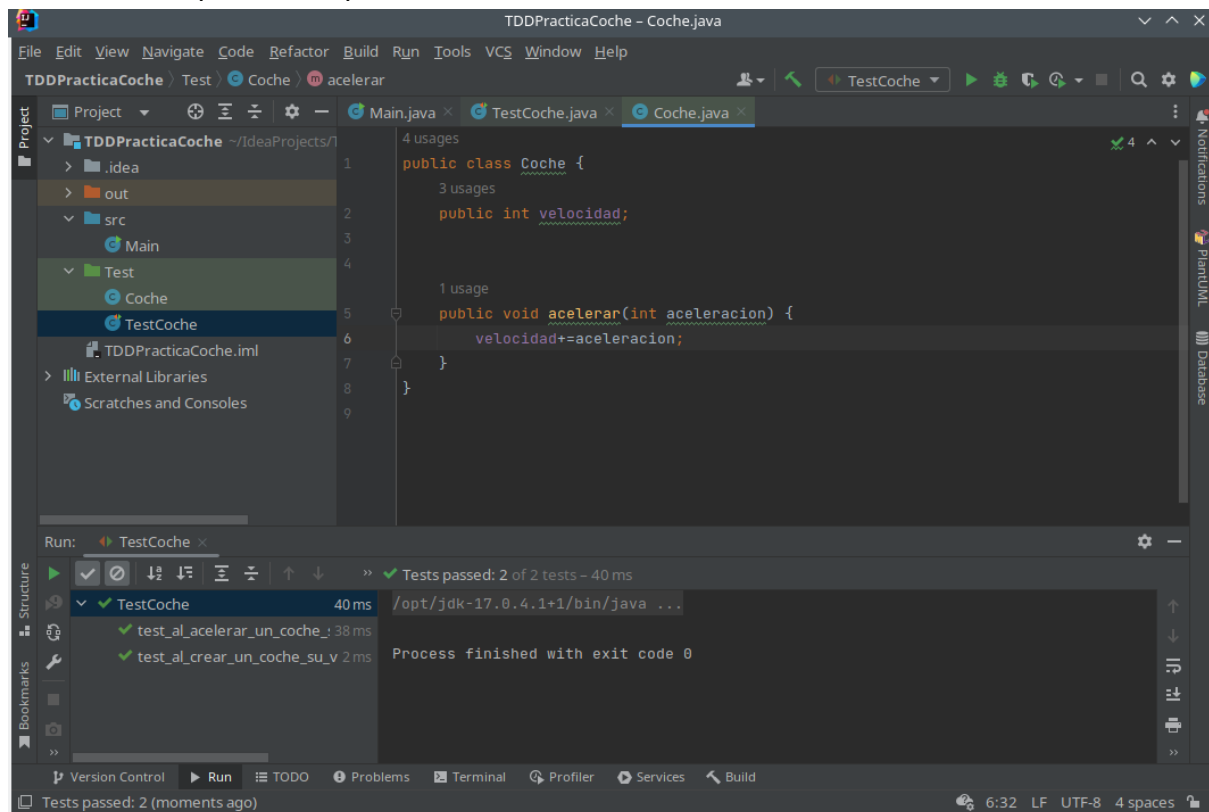
Se creará de esta forma.



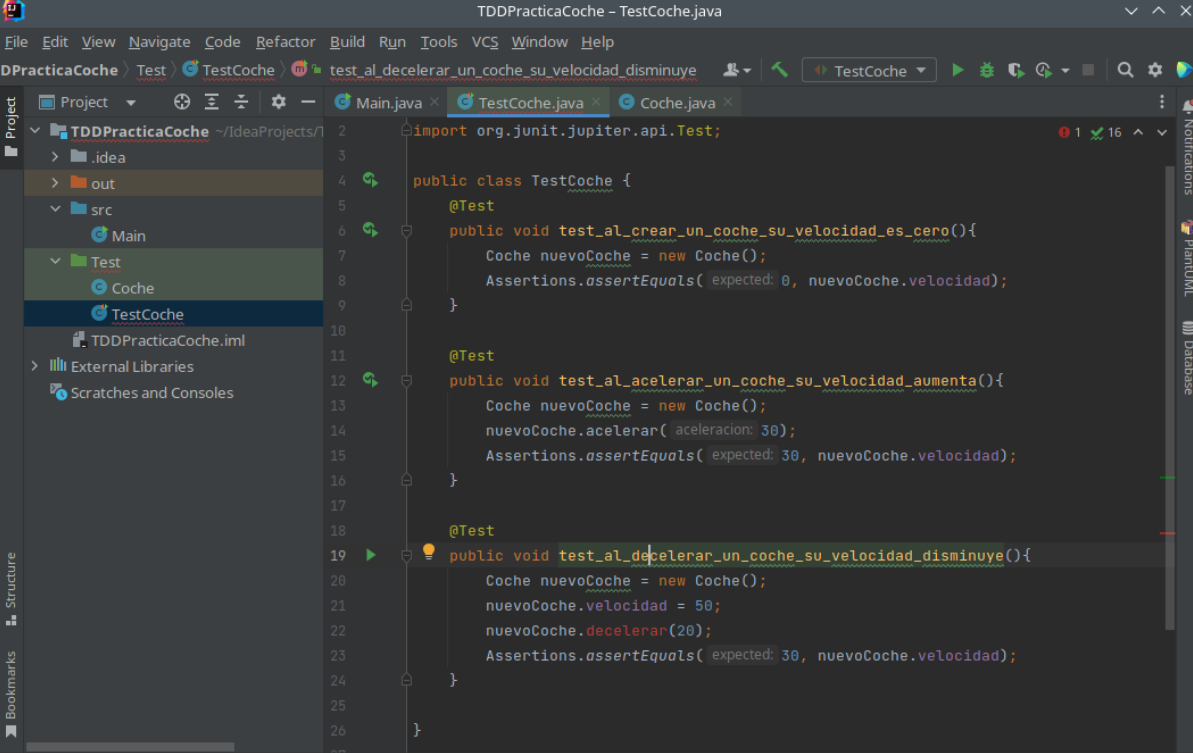
Y habrá que escribir que cuando se le llamé aumentará la velocidad con la línea `velocidad+=aceleracion`, siendo `aceleracion` el parámetro.



Y al realizar la prueba la aprobamos.



Creemos un tercer test, en este crearemos un coche, al que le asignaremos una velocidad de 50 y a continuación llamaremos al método decelerar del coche y por último, el método para comprobar la velocidad ahora tiene que comprobar que esta sea menor, por lo que si tiene una velocidad de 50 y por parámetro le decimos que disminuye en 20 tiene que comprobar que la velocidad del coche sea ahora de 30.



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

public class TestCoche {

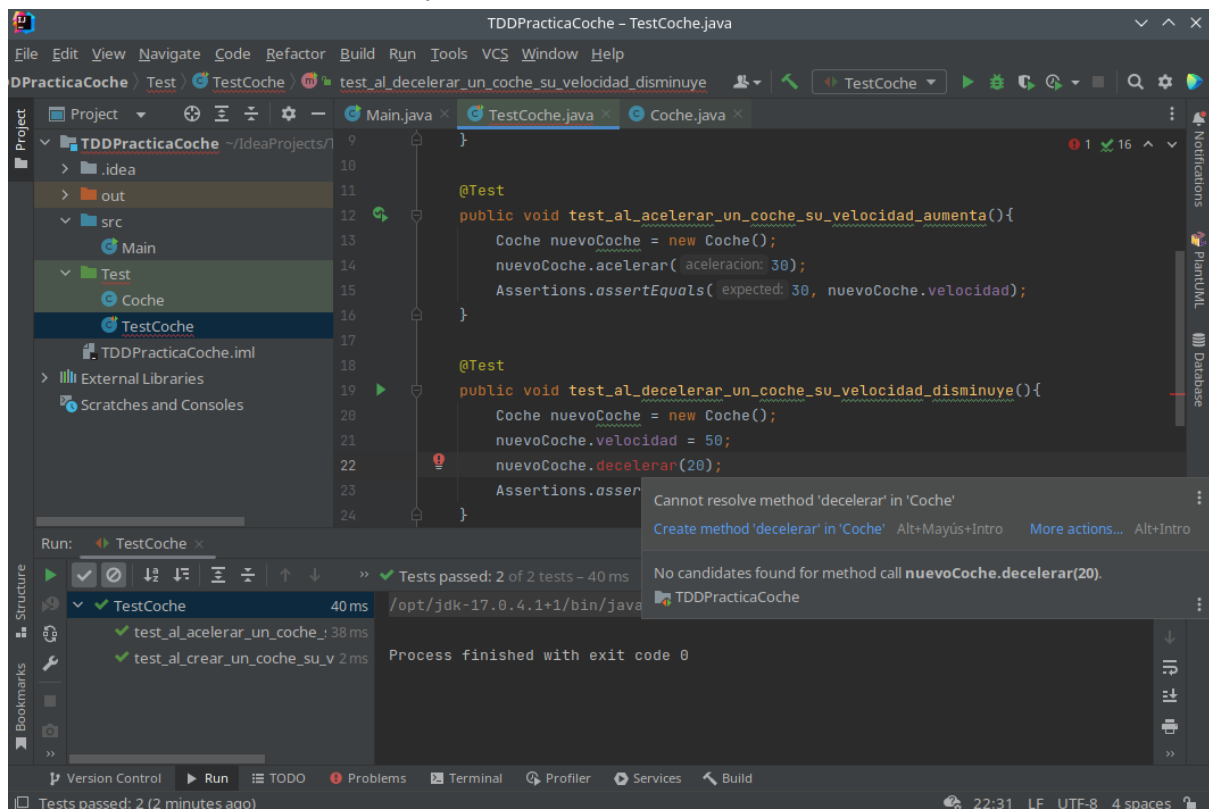
    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
    }

    @Test
    public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.acelerar( aceleracion: 30);
        Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
    }

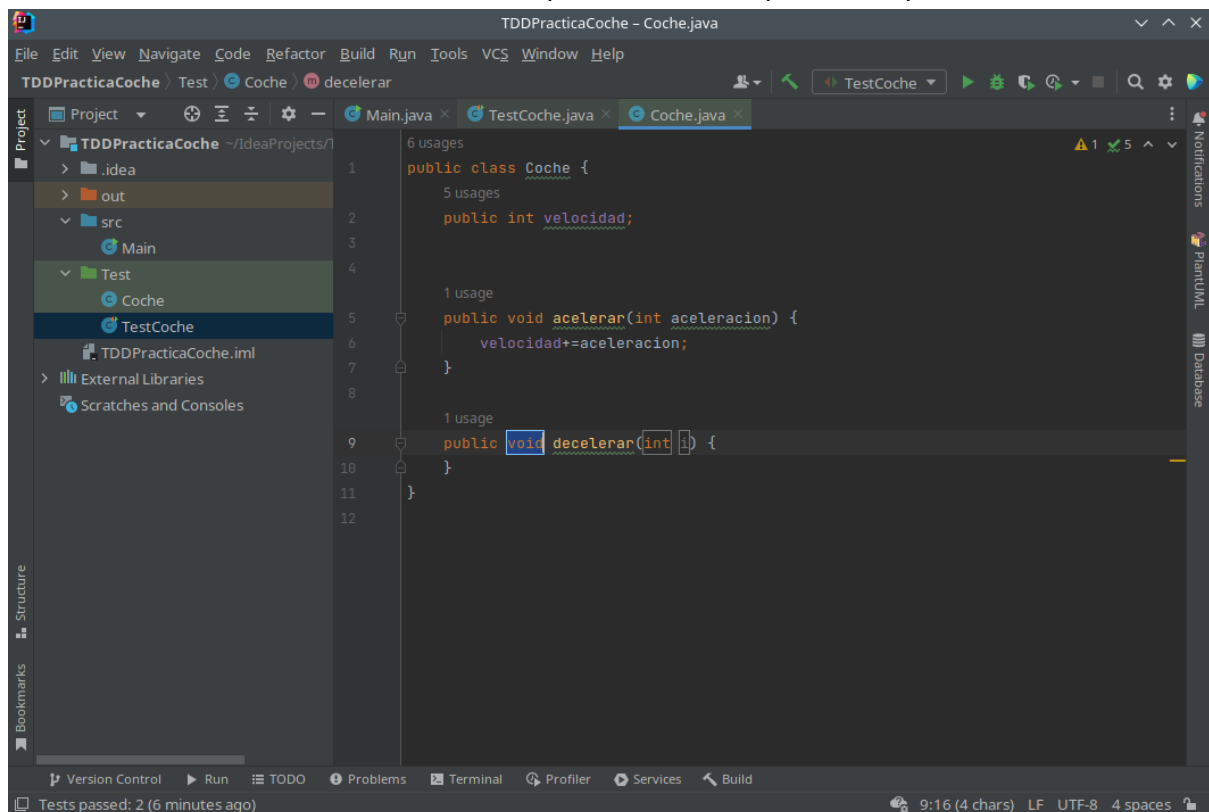
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_disminuye(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad = 50;
        nuevoCoche.decelerar(20);
        Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
    }
}
```

Typo: In word 'decelerar'

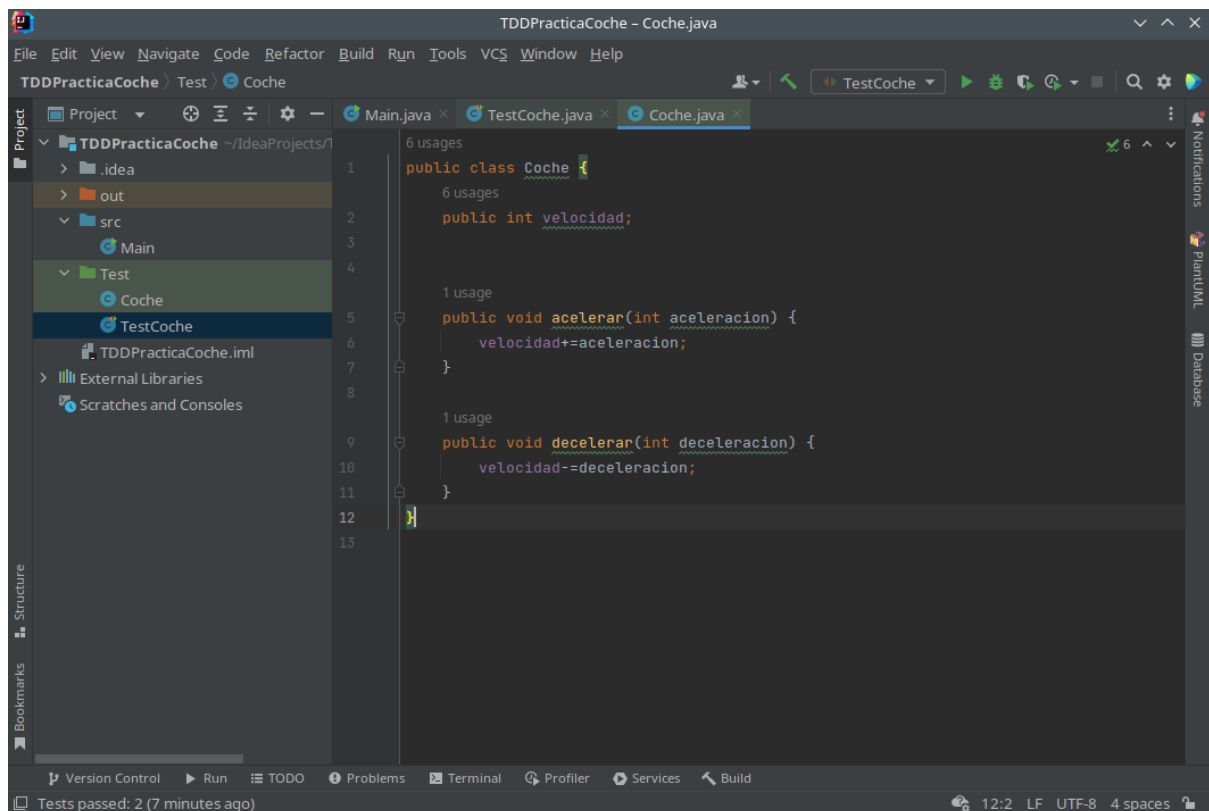
Como podemos observar el método decelerar de la clase Coche está en rojo porque no existe así que lo creamos como ya sabemos.



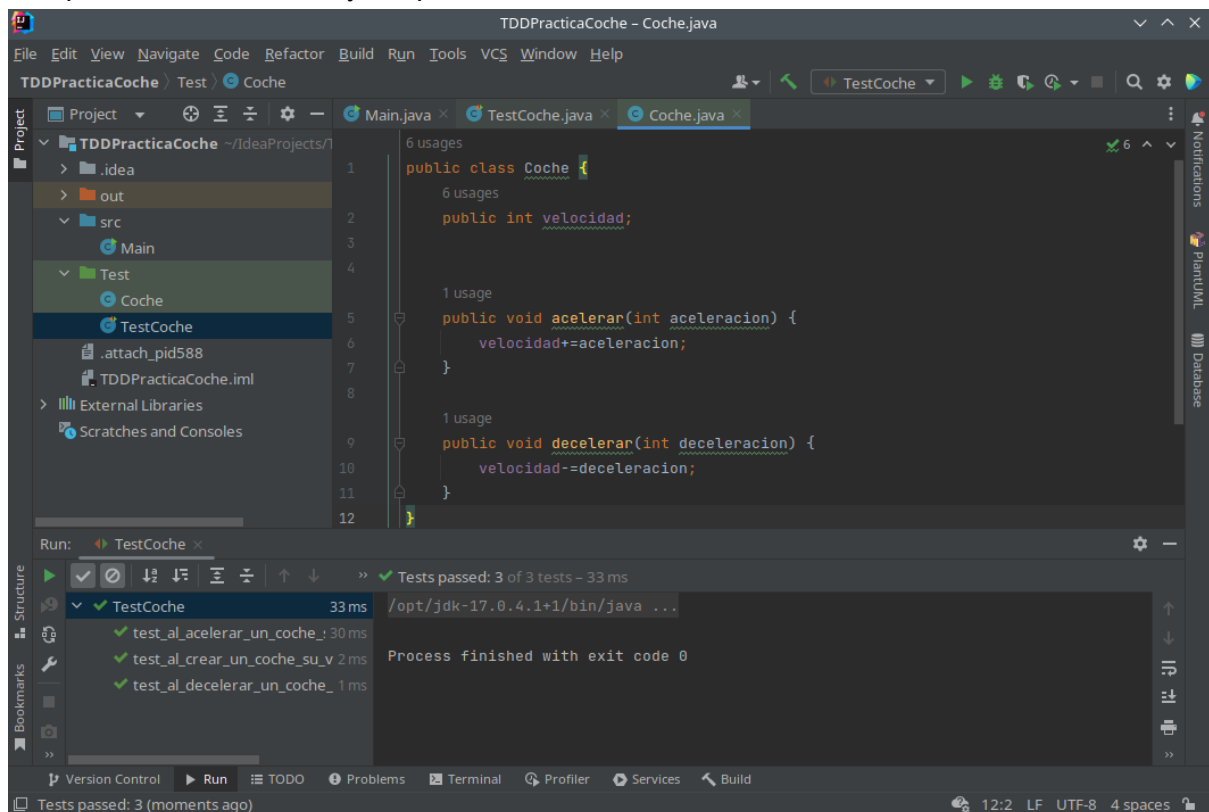
Al crearlo nos lo crea de la misma forma que acelerar así que habrá que modificarlo.



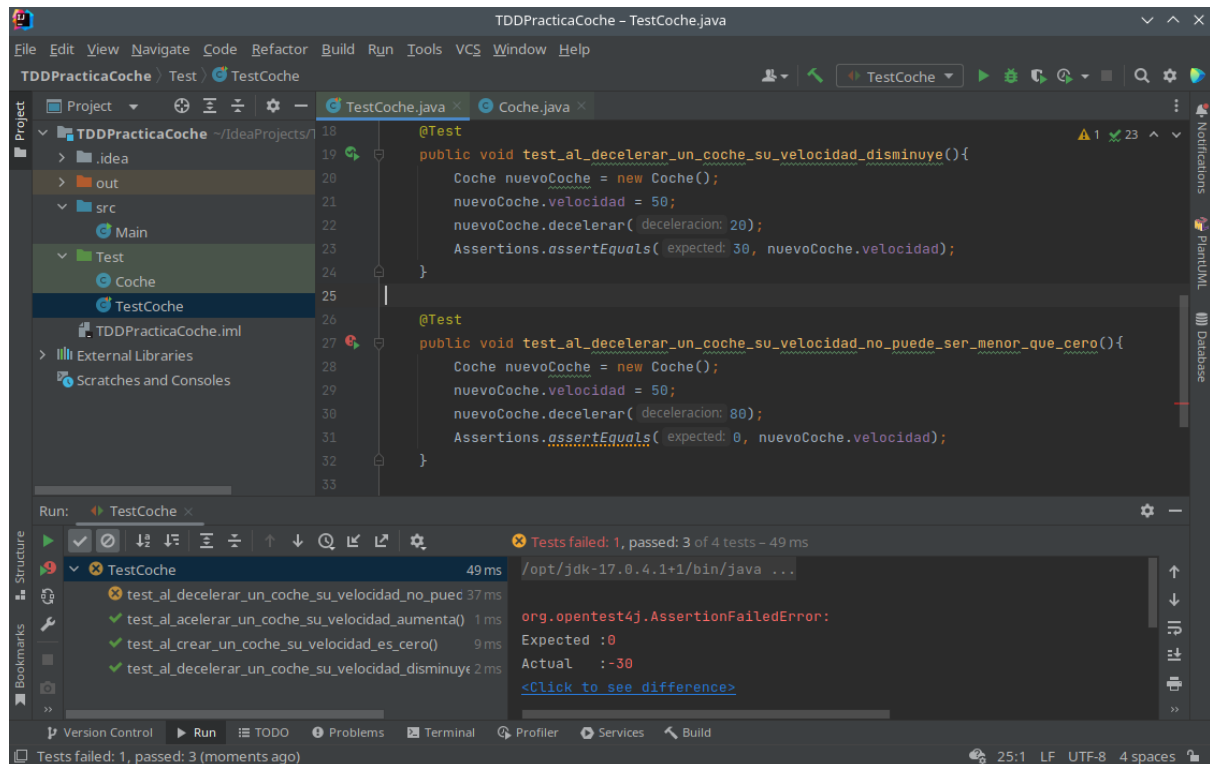
Con esta modificación, cuando se llame al método decelerar disminuirá la velocidad del coche.



Comprobamos con el test y lo aprobamos.



A continuación vamos a crear un cuarto y último test, este se ocupará de que a la hora de decelerar el coche la velocidad de este no pueda ser menor que cero. Para ello, copiamos el mismo test que hemos hecho antes y cambiamos como parámetro de 20 a 80 y el método que comprueba que la velocidad no sea menor que cero tenemos que cambiar el 30 por 0. Si hacemos ahora el test no lo pasaremos y en la consola podemos ver que espera un cero (Expected : 0) pero le ha llegado un -30 (Actual : -30) esto quiere decir que el método decelerar no controla que la velocidad no pueda ser menor que cero y hay que modificarlo.



The screenshot shows the IntelliJ IDEA IDE with a project named 'TDDPracticaCoche'. The 'Test' directory is selected in the Project view. The 'TestCoche.java' file is open, showing two test methods. The first test, 'test\_al\_decelerar\_un\_coche\_su\_velocidad\_disminuye()', is passing. The second test, 'test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_ser\_menor\_que\_cero()', is failing. The Run window shows the test results: 1 test failed, 3 passed. The failed test is 'test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_ser\_menor\_que\_cero()' with an error message: 'org.opentest4j.AssertionFailedError: Expected : 0 Actual : -30'. The console output shows the error details: 'Expected : 0', 'Actual : -30', and a link to see the difference.

```
18 @Test
19 public void test_al_decelerar_un_coche_su_velocidad_disminuye(){
20     Coche nuevoCoche = new Coche();
21     nuevoCoche.velocidad = 50;
22     nuevoCoche.decelerar( deceleracion: 20);
23     Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
24 }
25
26 @Test
27 public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
28     Coche nuevoCoche = new Coche();
29     nuevoCoche.velocidad = 50;
30     nuevoCoche.decelerar( deceleracion: 80);
31     Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
32 }
33
```

Run: TestCoche x

Tests failed: 1, passed: 3 of 4 tests - 49 ms

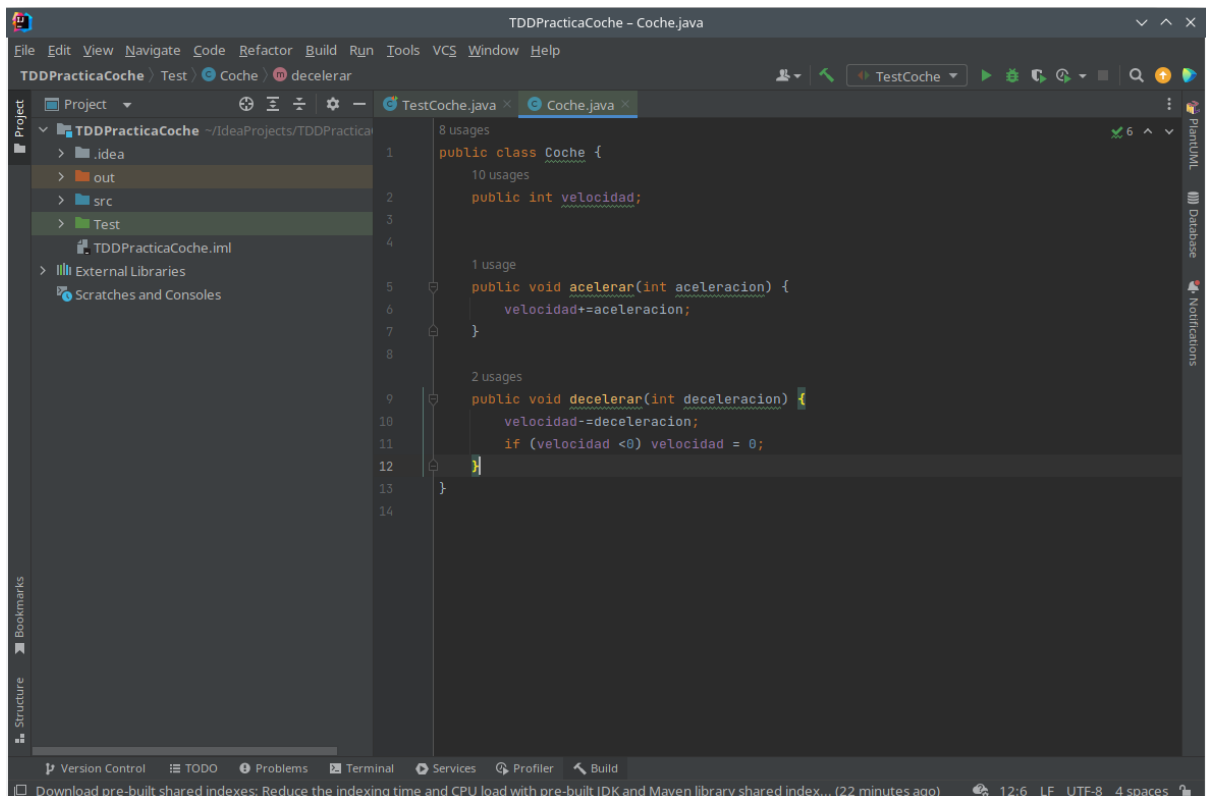
TestCoche

- test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_ser\_menor\_que\_cero() 37 ms
- test\_al\_acelerar\_un\_coche\_su\_velocidad\_aumenta() 1 ms
- test\_al\_crear\_un\_coche\_su\_velocidad\_es\_cero() 9 ms
- test\_al\_decelerar\_un\_coche\_su\_velocidad\_disminuye() 2 ms

org.opentest4j.AssertionFailedError:  
Expected : 0  
Actual : -30  
<Click to see difference>

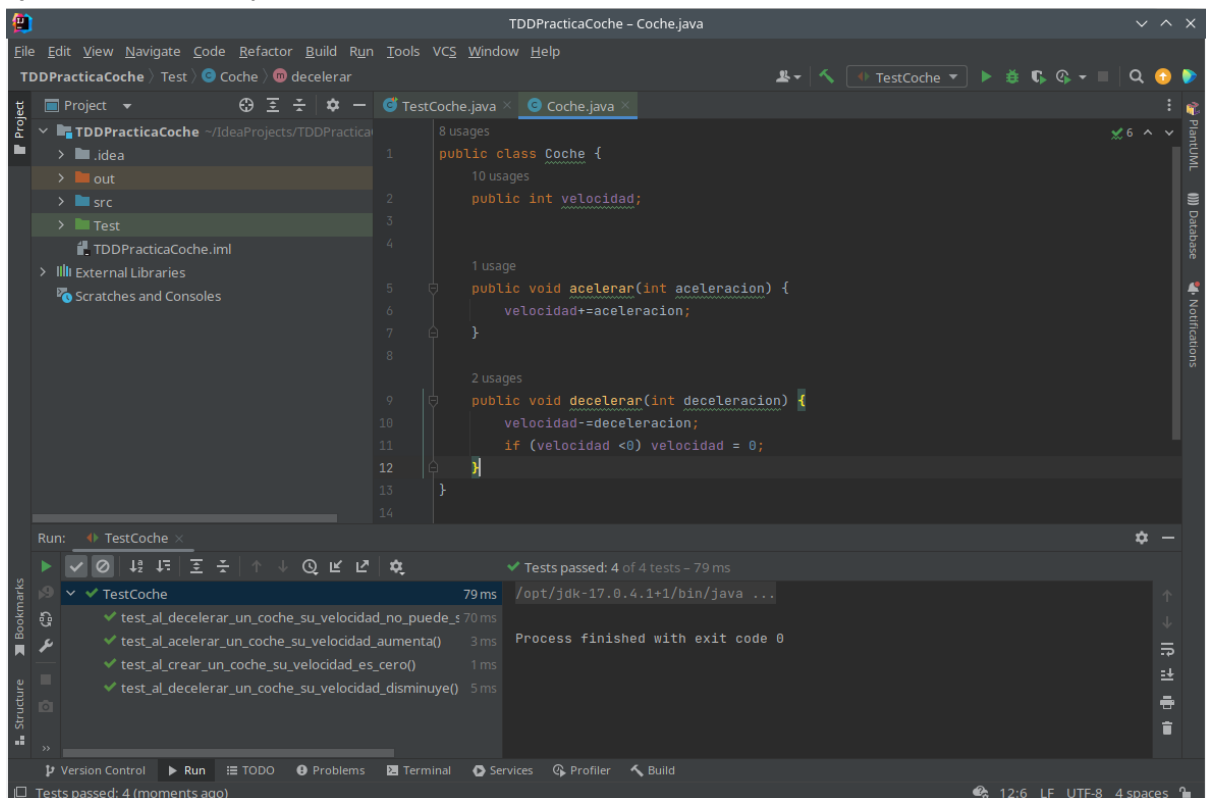


Modificamos el método decelerar de la clase Coche y añadimos una condición if en la que si la velocidad es menor que 0, la velocidad del coche será de 0.



```
1 public class Coche {
2     public int velocidad;
3
4
5     1 usage
6     public void acelerar(int aceleracion) {
7         velocidad+=aceleracion;
8     }
9
10    2 usages
11    public void decelerar(int deceleracion) {
12        velocidad-=deceleracion;
13        if (velocidad < 0) velocidad = 0;
14    }
15 }
```

Ejecutamos el test y ahora deberíamos pasarlo sin problemas.



```
1 public class Coche {
2     public int velocidad;
3
4
5     1 usage
6     public void acelerar(int aceleracion) {
7         velocidad+=aceleracion;
8     }
9
10    2 usages
11    public void decelerar(int deceleracion) {
12        velocidad-=deceleracion;
13        if (velocidad < 0) velocidad = 0;
14    }
15 }
```

Run: TestCoche x

Tests passed: 4 of 4 tests - 79 ms

- test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_s 70 ms
- test\_al\_acelerar\_un\_coche\_su\_velocidad\_aumenta() 3 ms
- test\_al\_crear\_un\_coche\_su\_velocidad\_es\_cero() 1 ms
- test\_al\_decelerar\_un\_coche\_su\_velocidad\_disminuye() 5 ms

Process finished with exit code 0