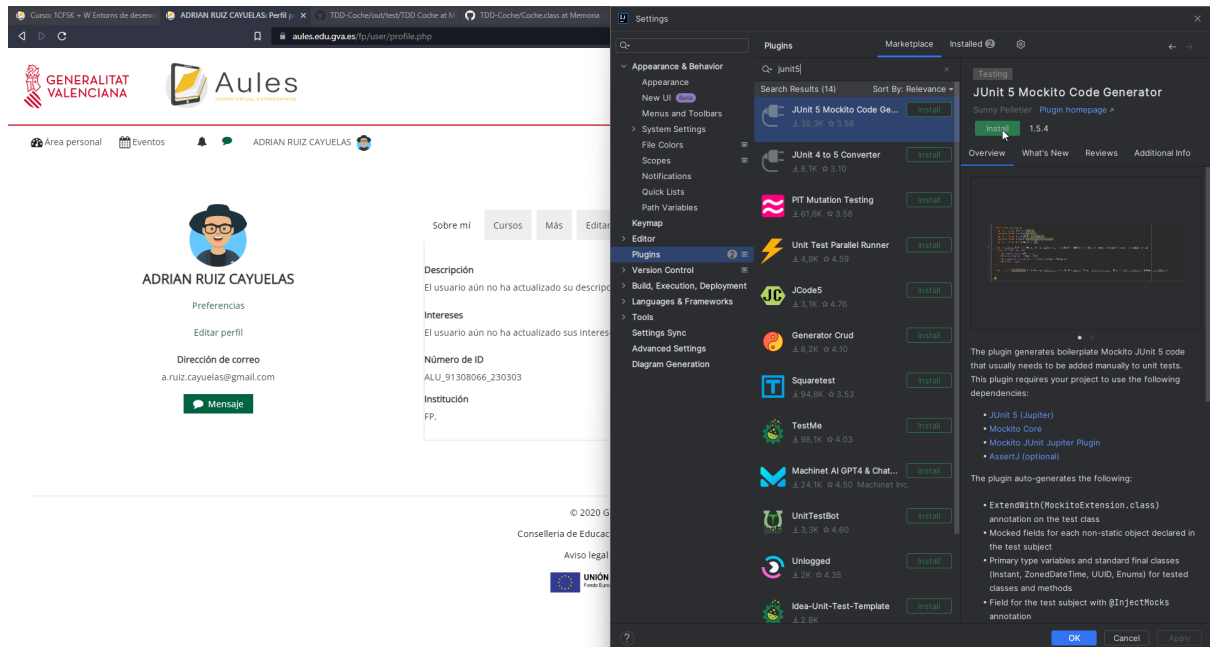


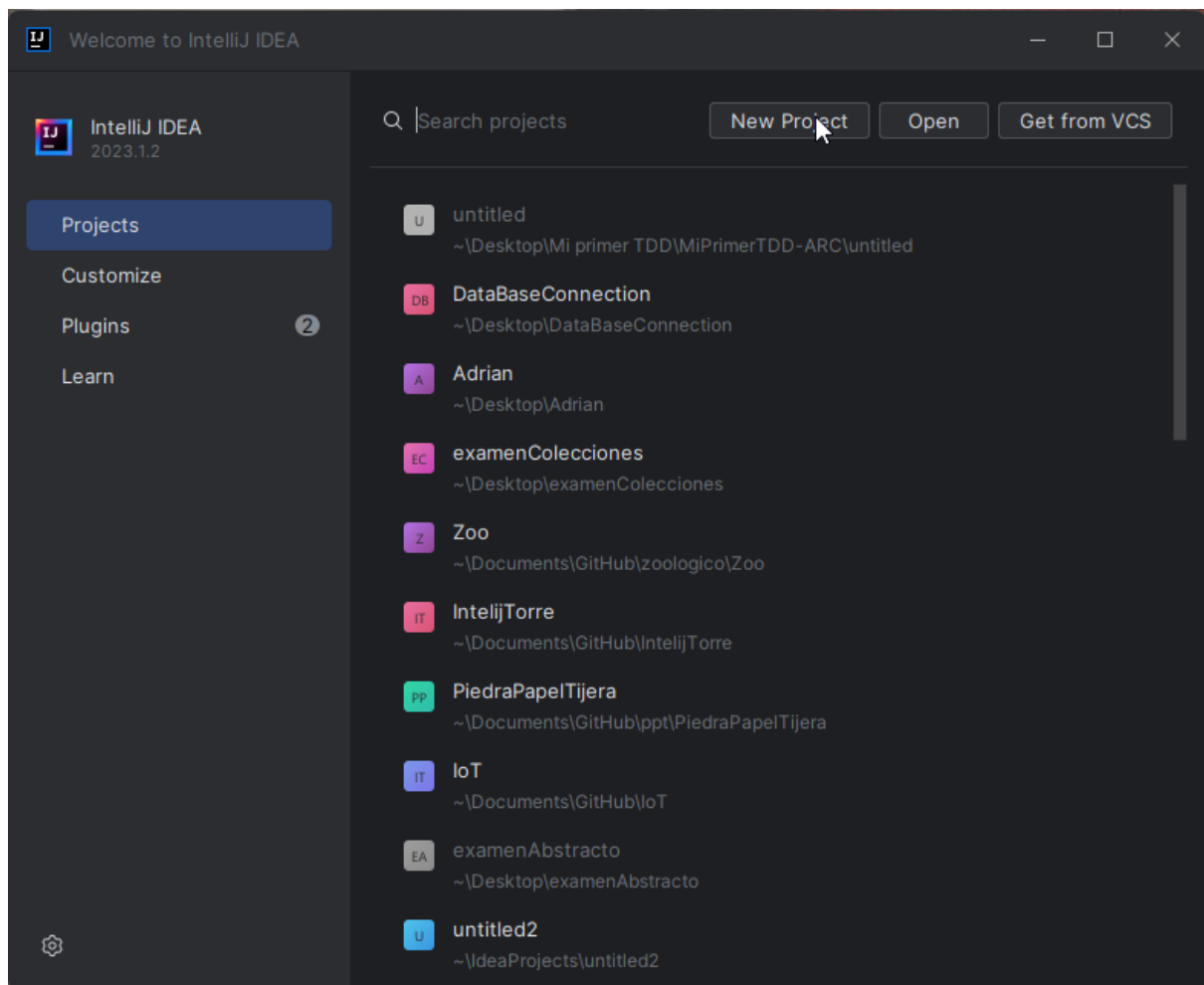
Memoria TDD IntelliJ Junit5

Antes de comenzar la practica deberemos de instalar el plugin de Junit5 :

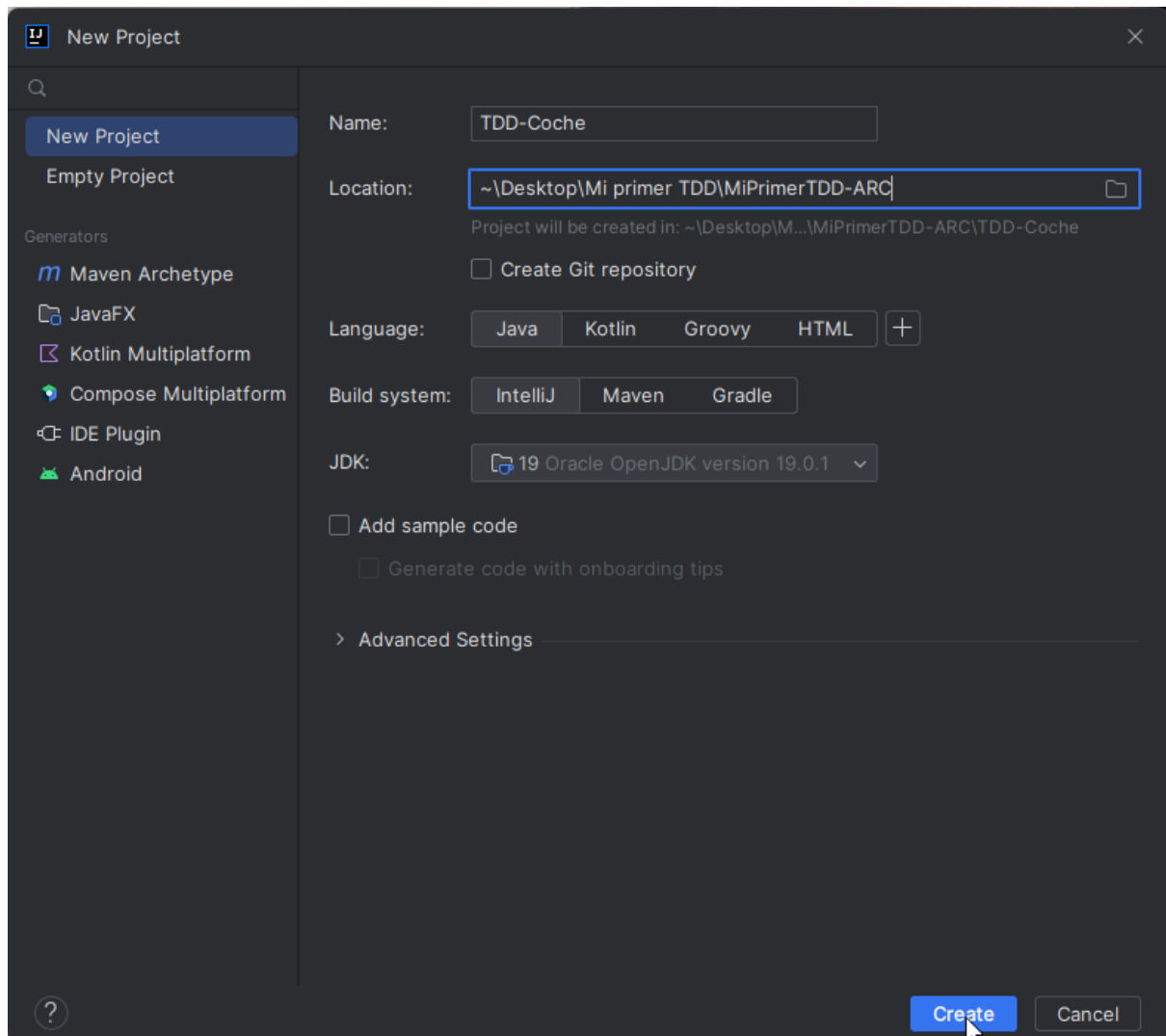


Tras esto crearemos un nuevo proyecto en IntelliJ, que yo en mi caso llamare **TDD-Coche**

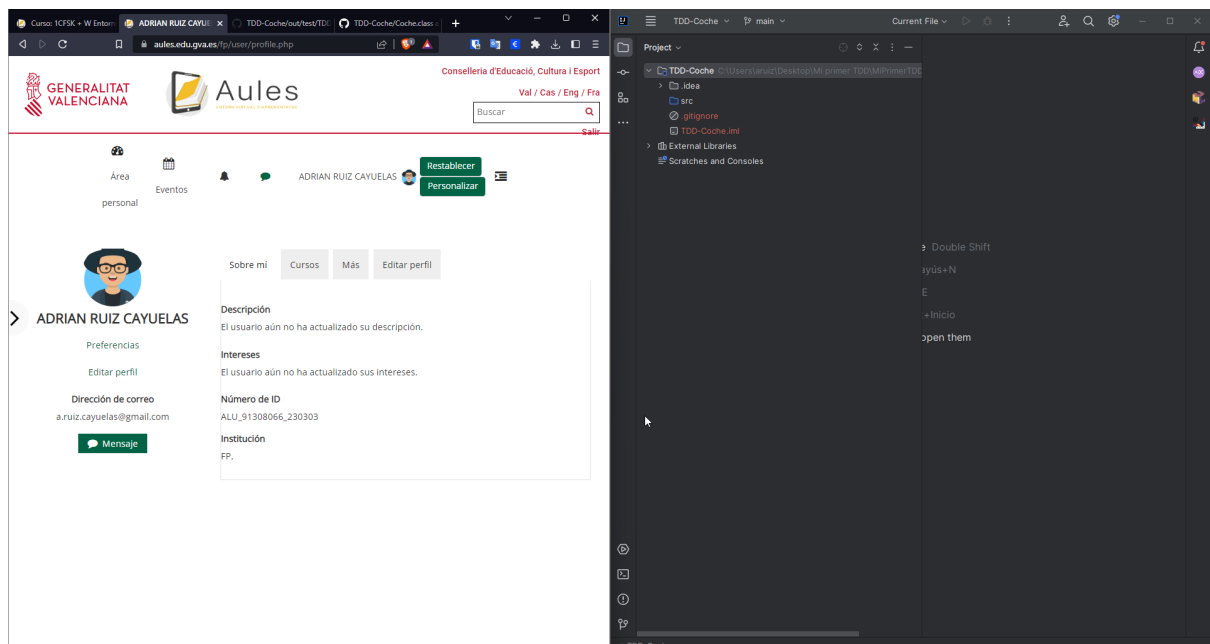
Para abriremos IntelliJ y haremos click en new project:



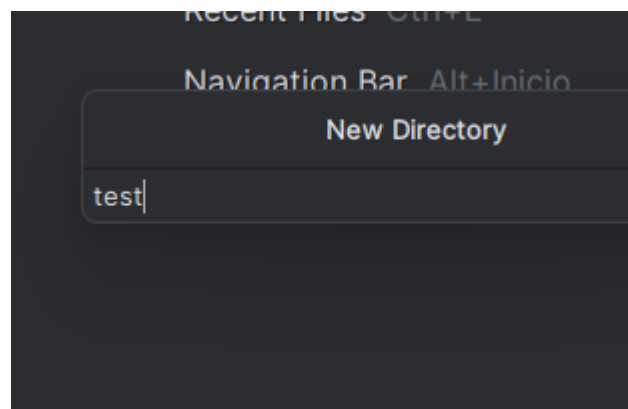
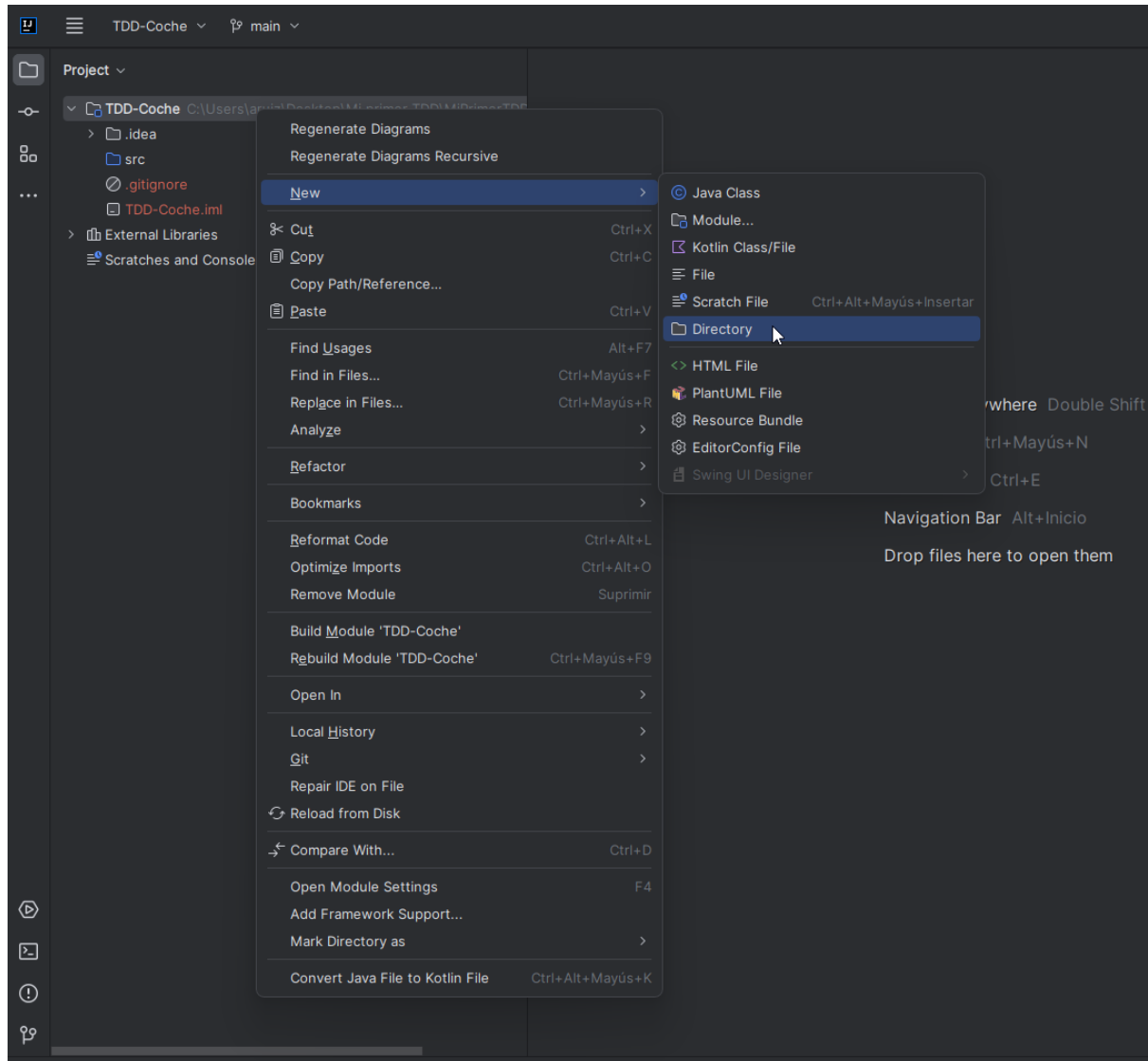
Rellenamos los datos para el nuevo proyecto y haremos click en Create:

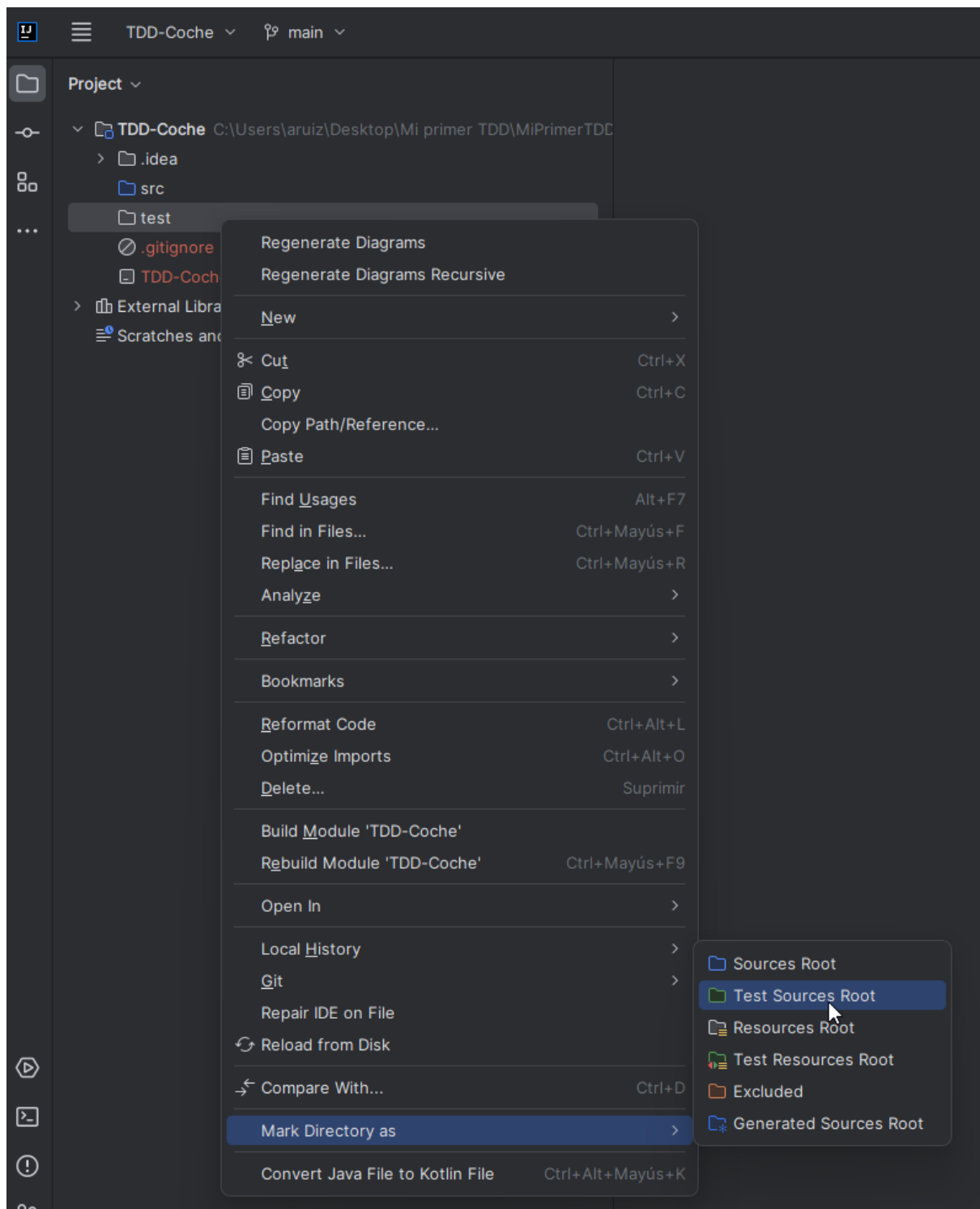


Con todo esto ya deberíamos de contar con un proyecto nuevo:

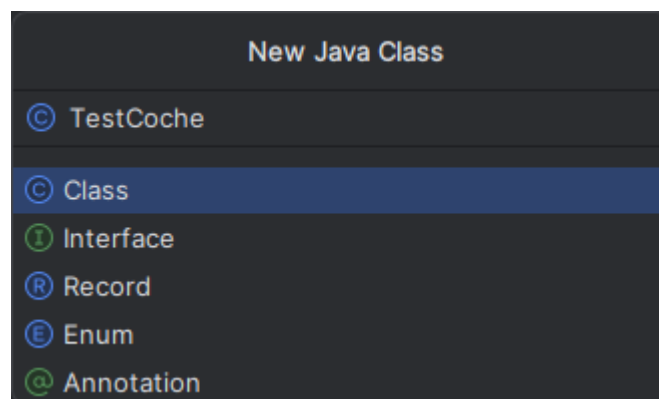
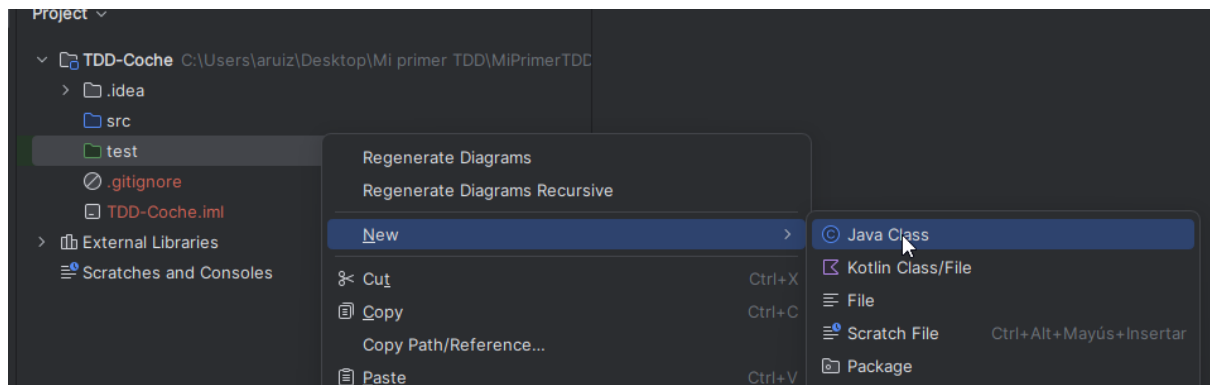


Ahora vamos a crear la carpeta de tests, para esto crearemos un directorio dentro del proyecto y lo marcaremos como directorio de tests:

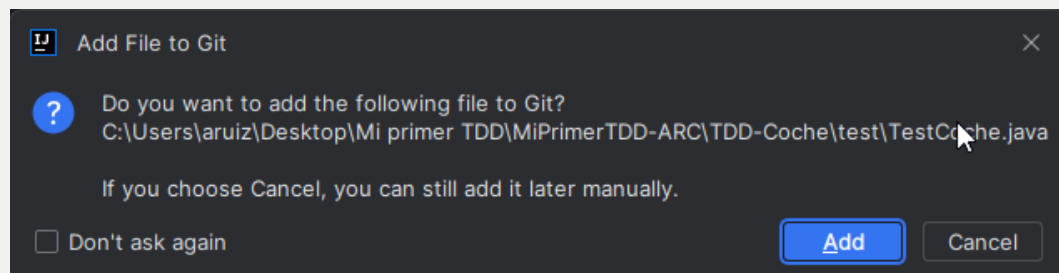




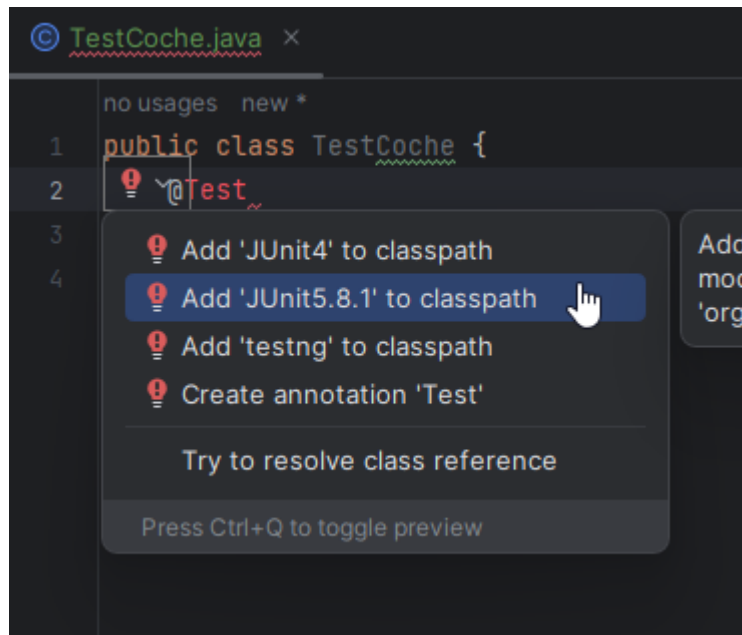
Ahora crearemos una clase java dentro del directorio creado anteriormente llamada **TestCoche**, indicaremos que se trata de un test usando la notación **@Test** e importaremos el paquete de Junit5:



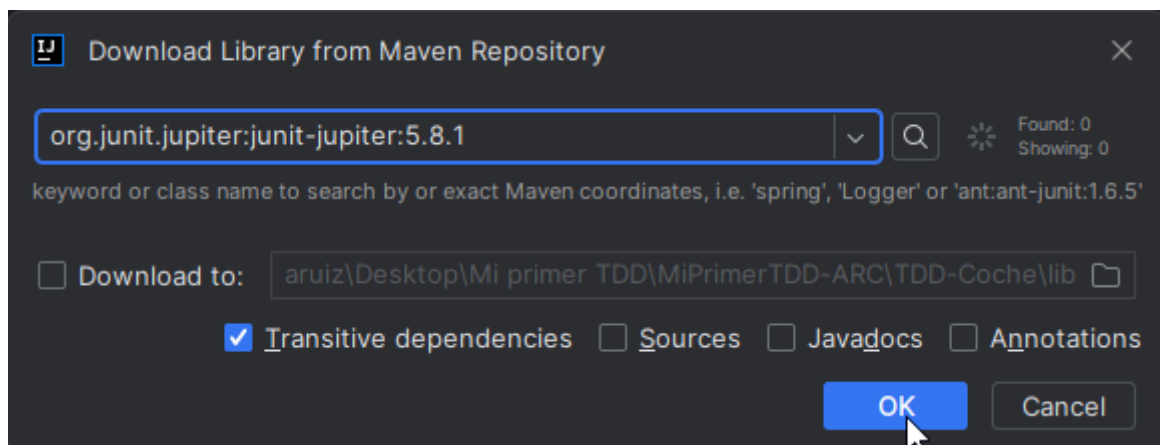
Si estamos trabajando con un VCS, como es el caso, se nos preguntará si deseamos añadir la clase java al repositorio con el siguiente mensaje:



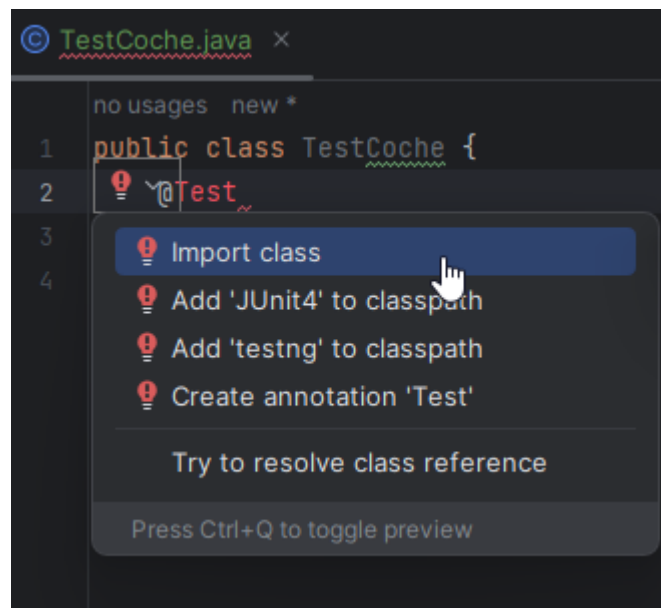
En mi caso la añadiré



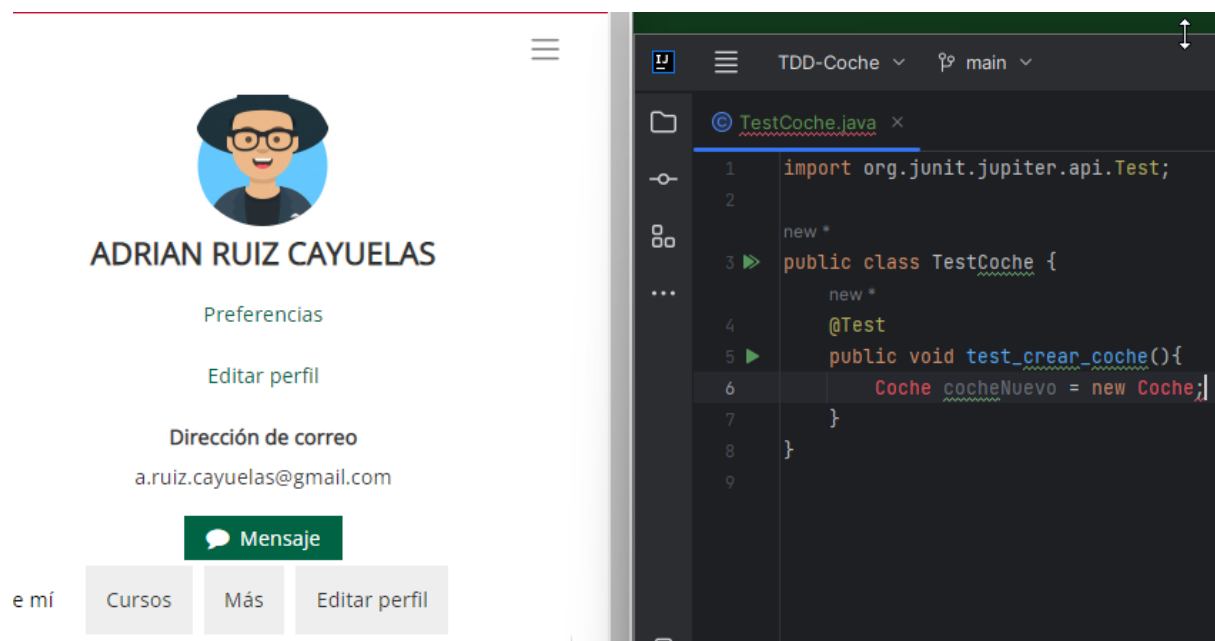
Ahora nos preguntará el IDE si queremos descargar el paquete del repositorio Maven, le diremos que si para poder continuar con el desarrollo de la práctica de manera normal:



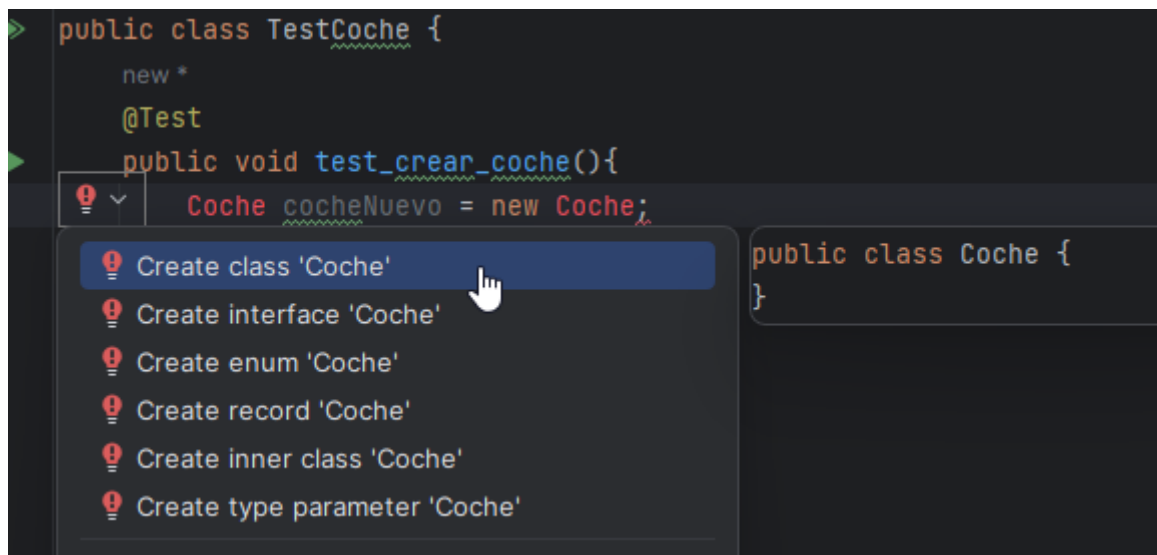
Tras esto solo restara importar la librería desde el menú contextual:



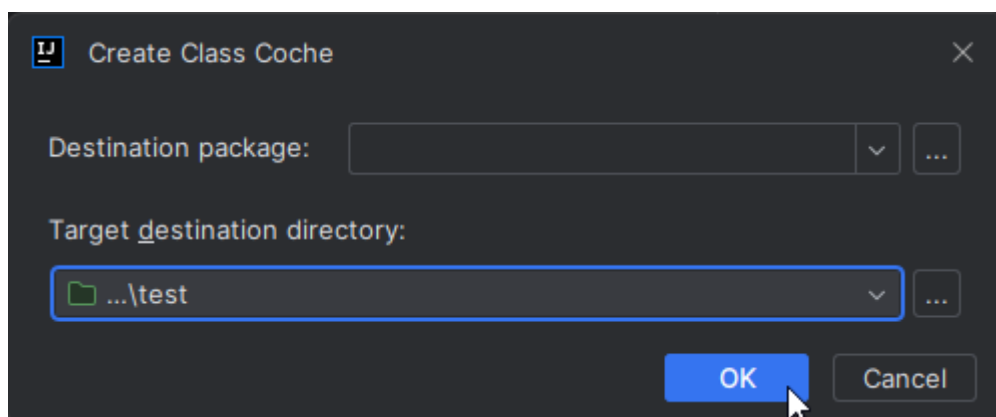
Ahora vamos a crear un método llamado **test-crear-coche** e instanciaremos un objeto de tipo coche, que llamaremos cocheNuevo:



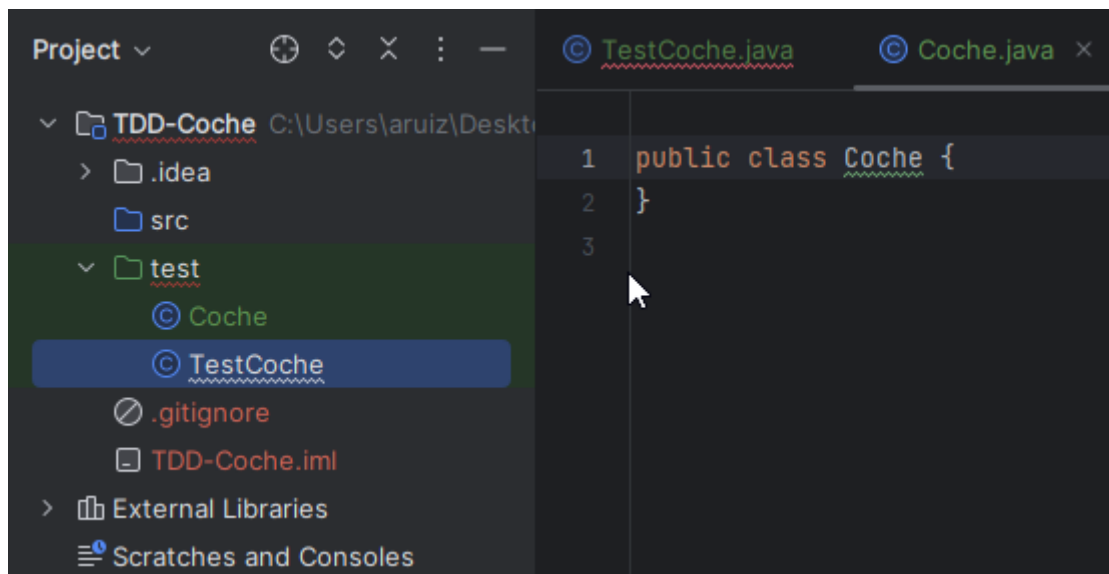
Puesto que la clase **Coche** no existe, el IDE no sabrá instanciar un objeto de ese tipo. Sin embargo, el IDE nos ofrece la posibilidad de crear por nosotros la clase, por tanto vamos a aceptar su sugerencia;



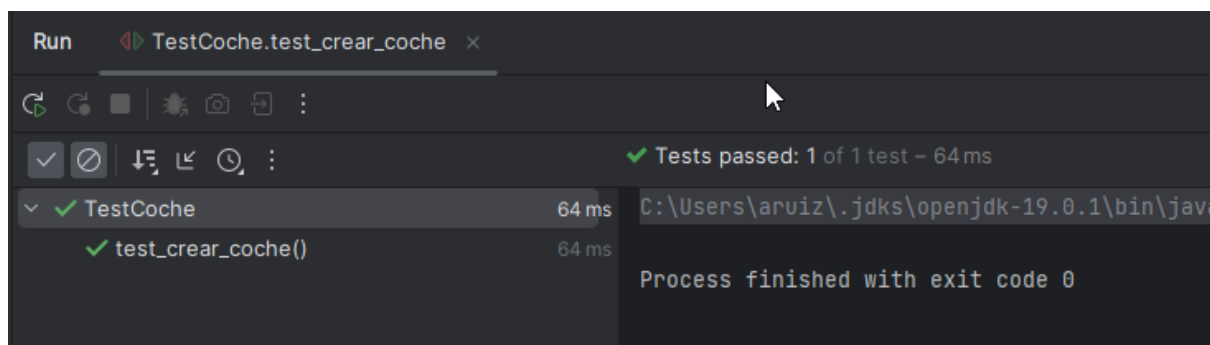
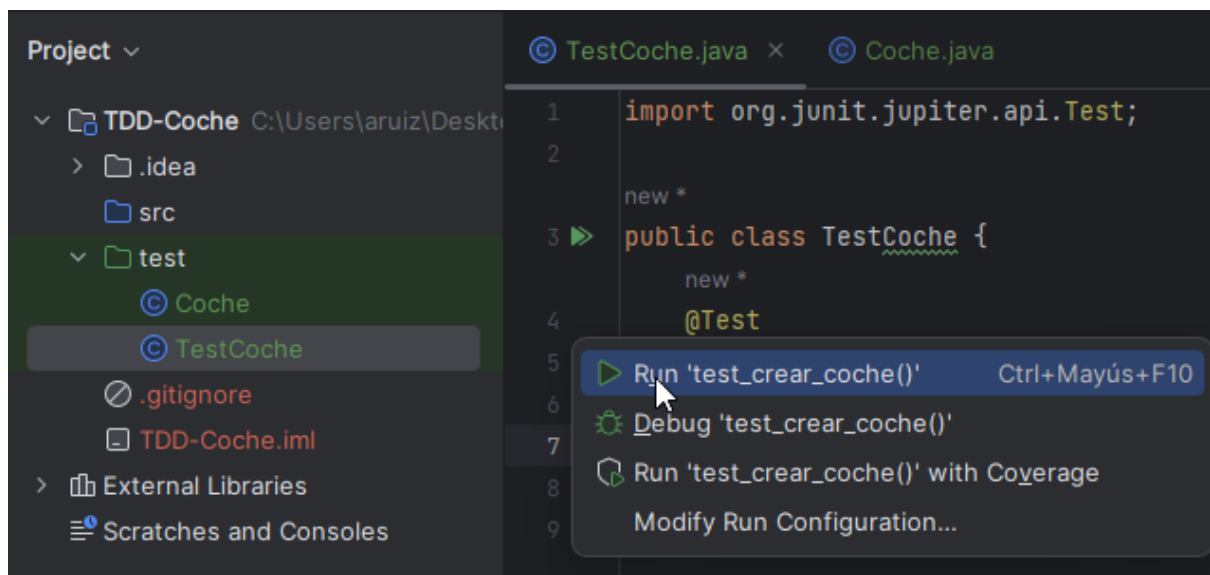
Ahora nos preguntará el destino para la clase, puesto que no queremos variar la destinación por defecto (que es la actual) simplemente aceptaremos:



Como podemos observar se ha creado de manera automática la clase:



Ahora probaremos a ejecutar el test, el cual debería de dar un resultado satisfactorio:

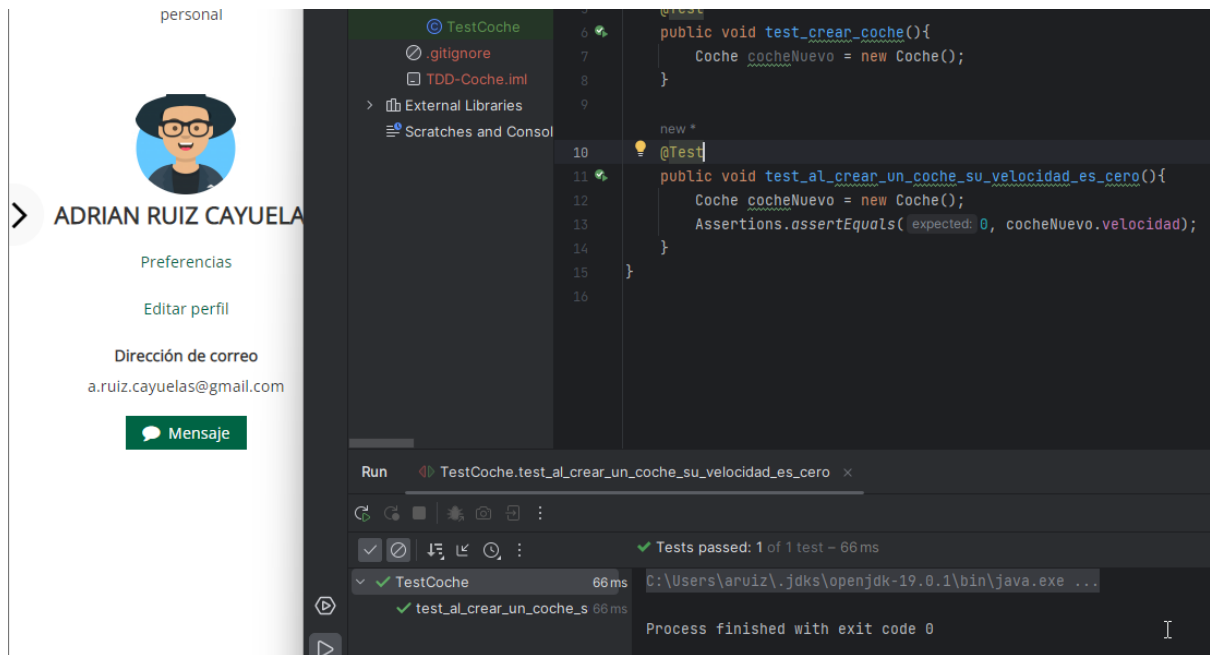


Ahora vamos a realizar un test de tipo **assertEquals** sobre el parámetro velocidad del coche. Como esta variable no existe el IDE nos sugerirá crearla:

Y si no vamos a la clase coche podremos comprobar que efectivamente se ha generado el atributo velocidad:



Ya solo nos resta ejecutar el test y comprobar que este se completa de manera satisfactoria:



Ahora crearemos un nuevo test y llamaremos al método acelerar pasándole como parámetro el valor 30. Al no existir el método mencionado el IDE nos sugerirá crearlo de manera automática, por lo cual aceptaremos:

Tras esto daremos cuerpo al método recién creado para que este cumpla la función que requerimos:





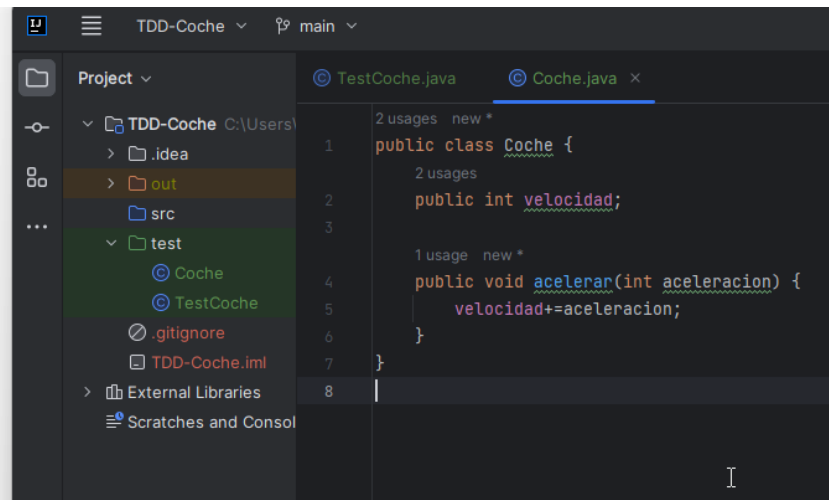
ADRIAN RUIZ CAYUELAS

Preferencias

Editar perfil

Dirección de correo
a.ruiz.cayuelas@gmail.com

Mensaje



The screenshot shows the IntelliJ IDEA interface. On the left, the 'Project' view displays the structure of the 'TDD-Coche' project, including folders like '.idea', 'out', 'src', and 'test', and files like 'Coche', 'TestCoche', '.gitignore', and 'TDD-Coche.iml'. The main editor shows the 'Coche.java' file with the following code:

```

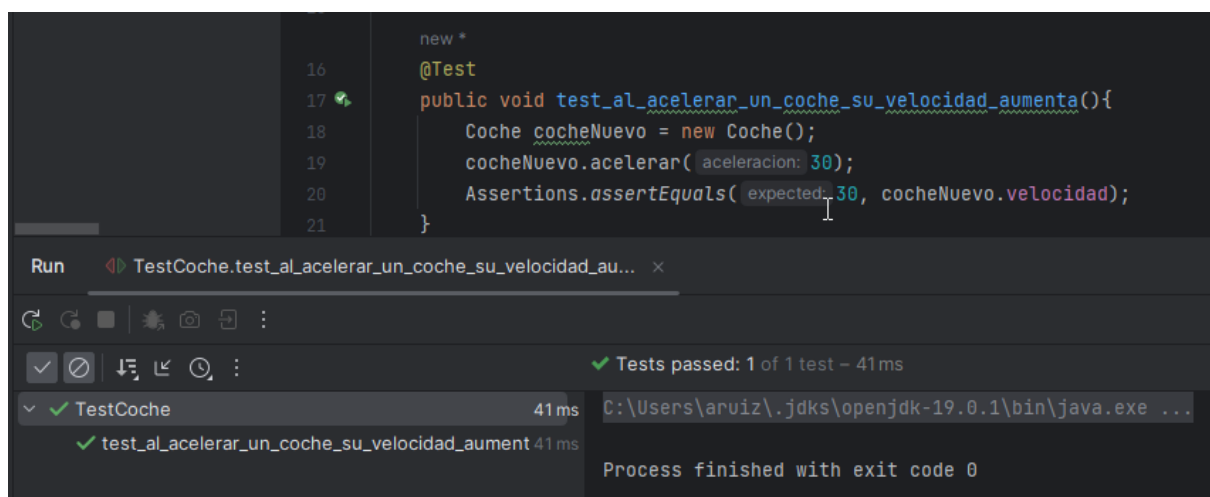
1 2 usages new *
2 public class Coche {
3     2 usages
4     public int velocidad;
5     1 usage new *
6     public void acelerar(int aceleracion) {
7         velocidad+=aceleracion;
8     }

```



Como se puede observar en las dos capturas anteriores el nombre del parámetro no coincide, esto es debido a que el nombre original no era para nada intuitivo y podía dar lugar a problemas de legibilidad cuando el software escale

Ahora crearemos un **Assertions.assertEquals** y testaremos que efectivamente la velocidad aumente en 30:



The screenshot shows the IntelliJ IDEA interface with a JUnit test being executed. The test code is as follows:

```

16 new *
17 @Test
18 public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
19     Coche cocheNuevo = new Coche();
20     cocheNuevo.acelerar( aceleracion: 30);
21     Assertions.assertEquals( expected: 30, cocheNuevo.velocidad);

```

The 'Run' button is clicked, and the test is executed. The output shows that the test passed successfully.

Run TestCoche.test_al_acelerar_un_coche_su_velocidad_au... x

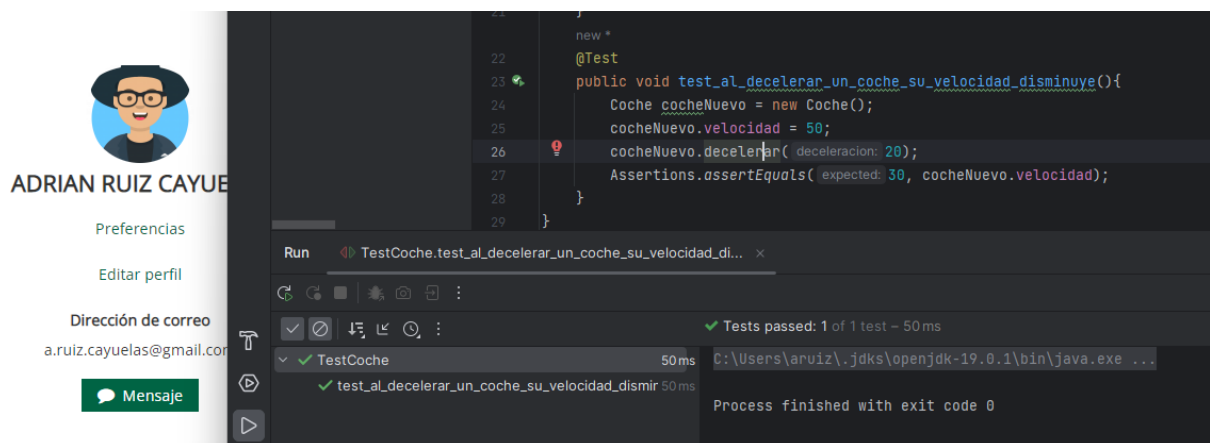
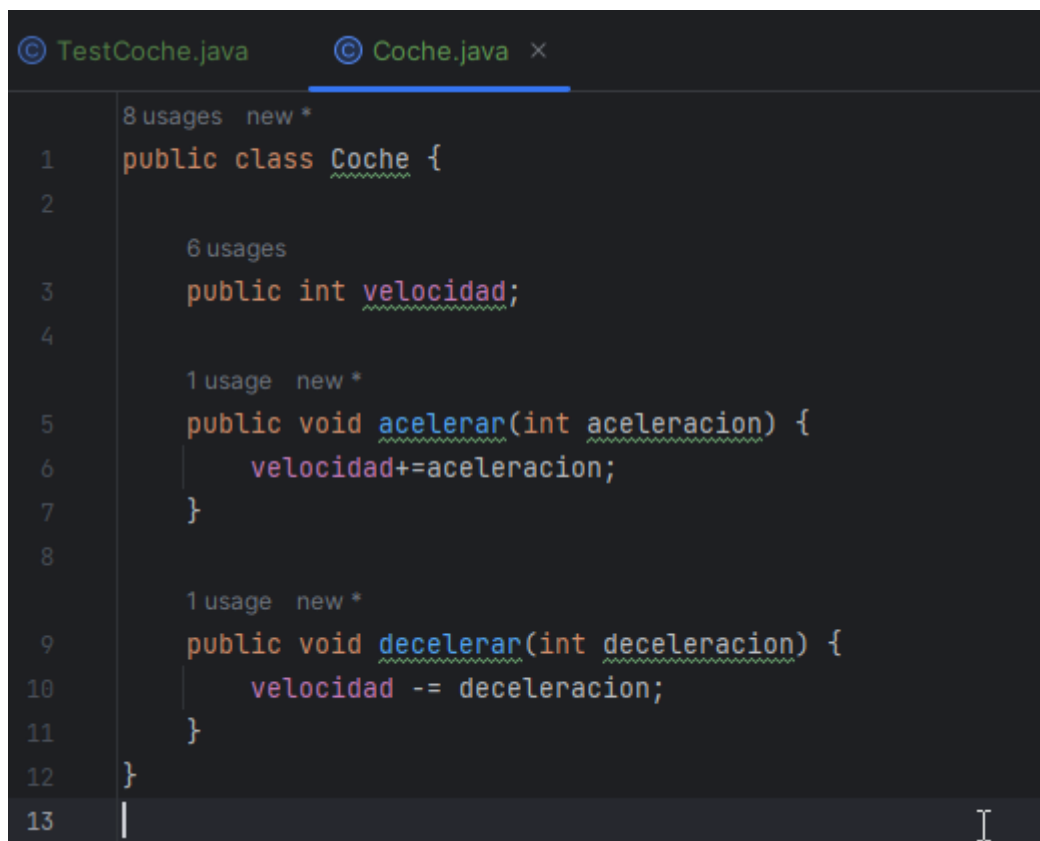
Tests passed: 1 of 1 test - 41ms

TestCoche 41ms

test_al_acelerar_un_coche_su_velocidad_aument 41ms

Process finished with exit code 0

Tras haber creado el método acelerar, procederemos a crear el de decelerar. Para ello llamaremos al método desde un nuevo test y replicaremos el proceso seguido en las capturas anteriores, pero adaptado a las necesidades de este nuevo método:





En el test se establece la velocidad inicialmente para que el test tenga lógica y se pueda comprobar de manera correcta

Para el siguiente test reutilizaremos el código del test anterior, pero esta vez probaremos que la velocidad mínima del coche no sea inferior a 0. Para esto solo deberemos ajustar los parámetros del test y ver que resultado obtenemos:

```
28     }
29     new *
30     @Test
31     public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
32         Coche cocheNuevo = new Coche();
33         cocheNuevo.velocidad = 50;
34         cocheNuevo.decelerar( deceleracion: 80);
35         Assertions.assertEquals( expected: 0, cocheNuevo.velocidad);
36     }
```

Run TestCoche.test_al_decelerar_un_coche_su_velocidad_n... x

Tests failed: 1 of 1 test - 86 ms

TestCoche 86 ms C:\Users\arviz\.jdk\openjdk-19.0.1\bin\java.exe ...

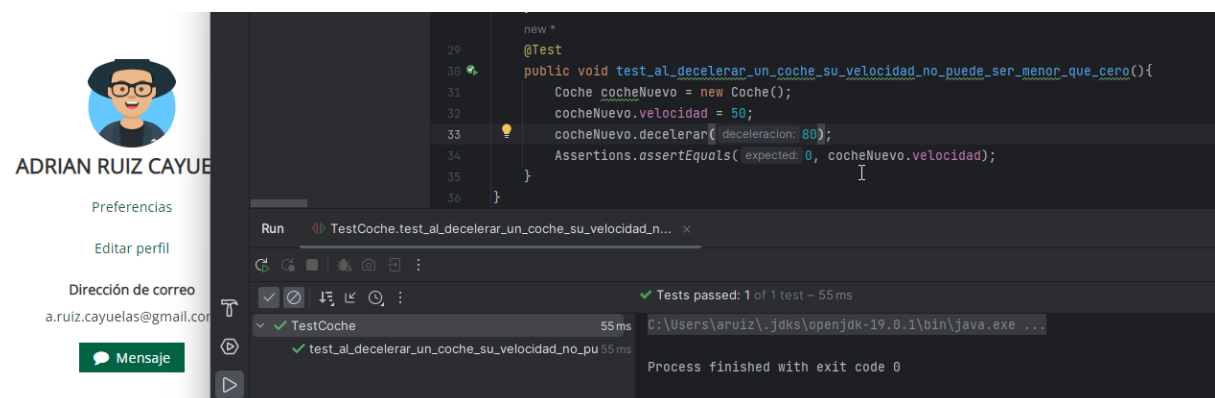
test_al_decelerar_un_coche_su_velocidad_no_pu 86 ms

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

Como podemos observar no pasamos el test, por tanto deberemos de ir a la clase coche y ajustar el método decelerar para que la condición se cumpla y el test sea satisfactorio:

```
TestCoche.java Coche.java x
10 usages new *
1 public class Coche {
2
3     10 usages
4     public int velocidad;
5
6     1 usage new *
7     public void acelerar(int aceleracion) {
8         velocidad+=aceleracion;
9     }
10
11     2 usages new *
12     public void decelerar(int deceleracion) {
13         velocidad -= deceleracion;
14         if (velocidad < 0) {
15             velocidad = 0;
16         }
17     }
18 }
```

Ahora volveremos a realizar el test y comprobaremos que ahora si que funcione de manera correcta:



Como podemos observar, ahora todos los tests tienen un resultado satisfactorio