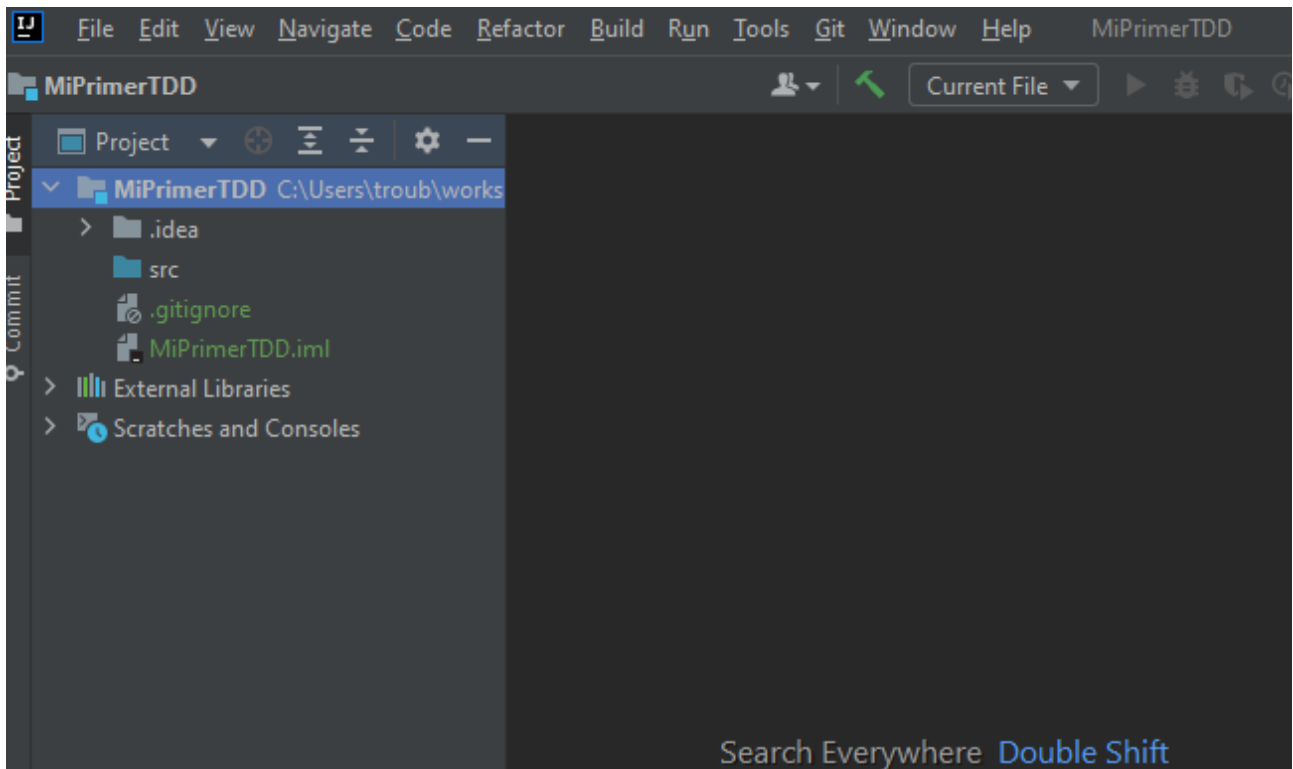
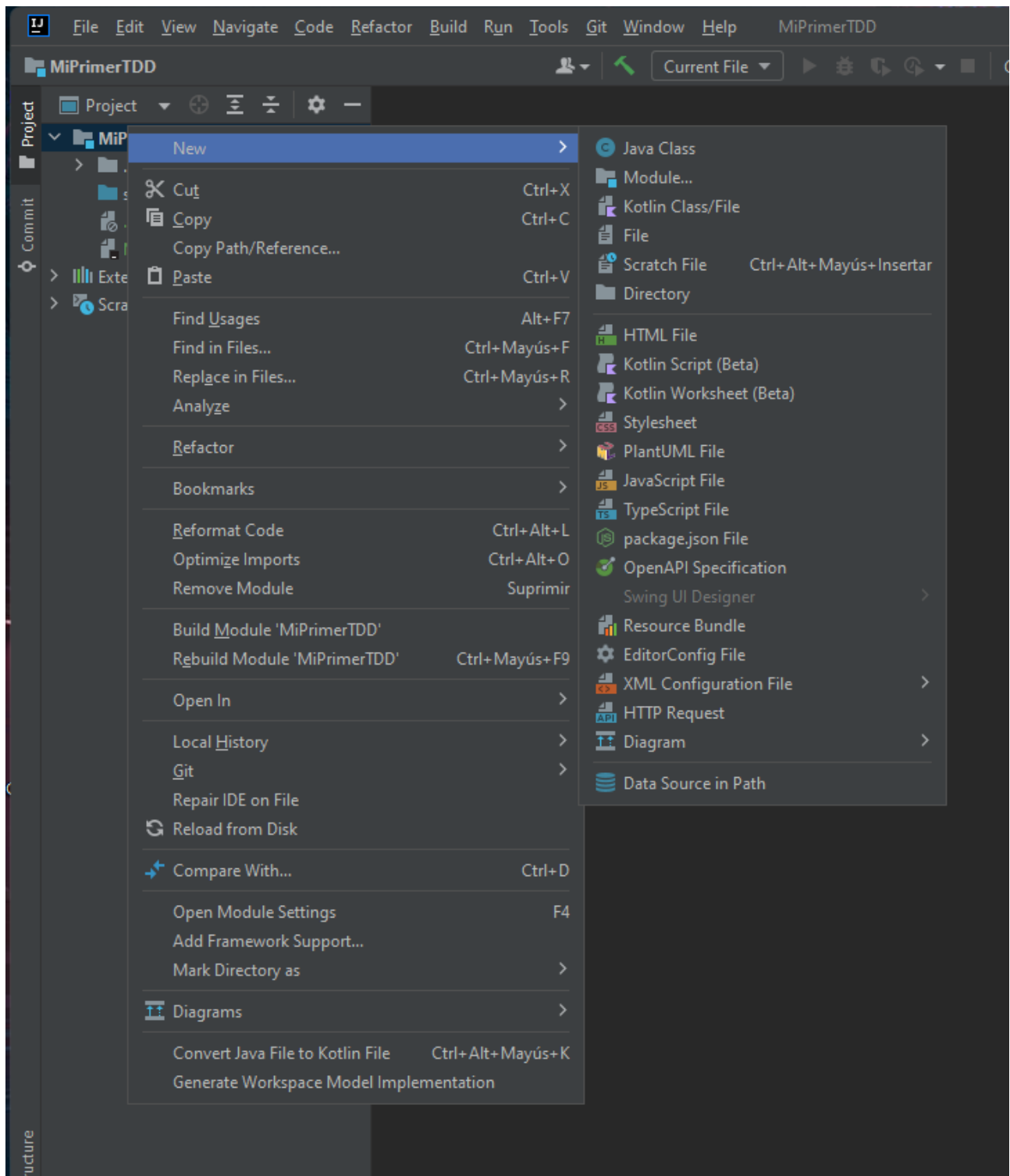


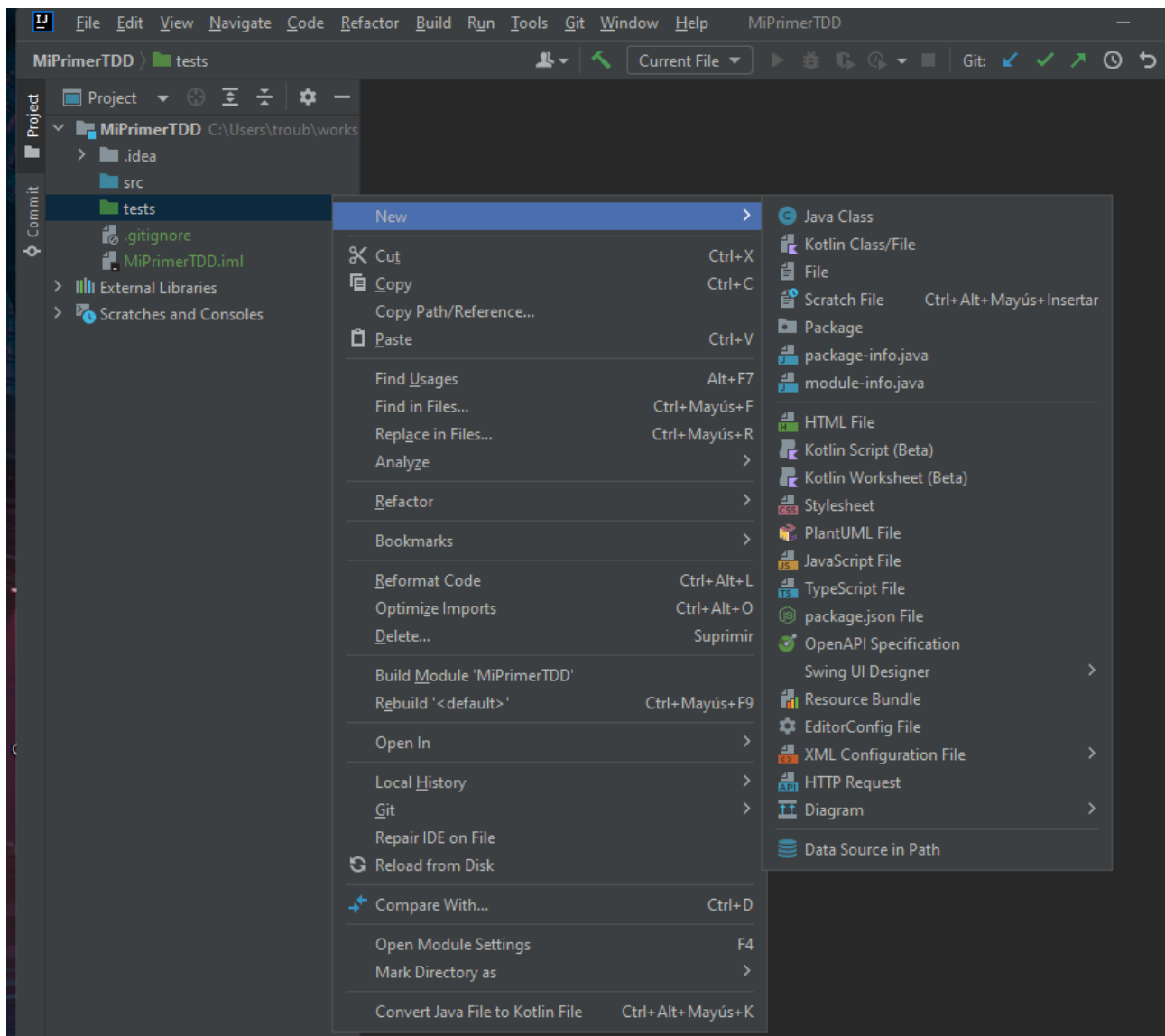
Creamos un proyecto en intelliJ →

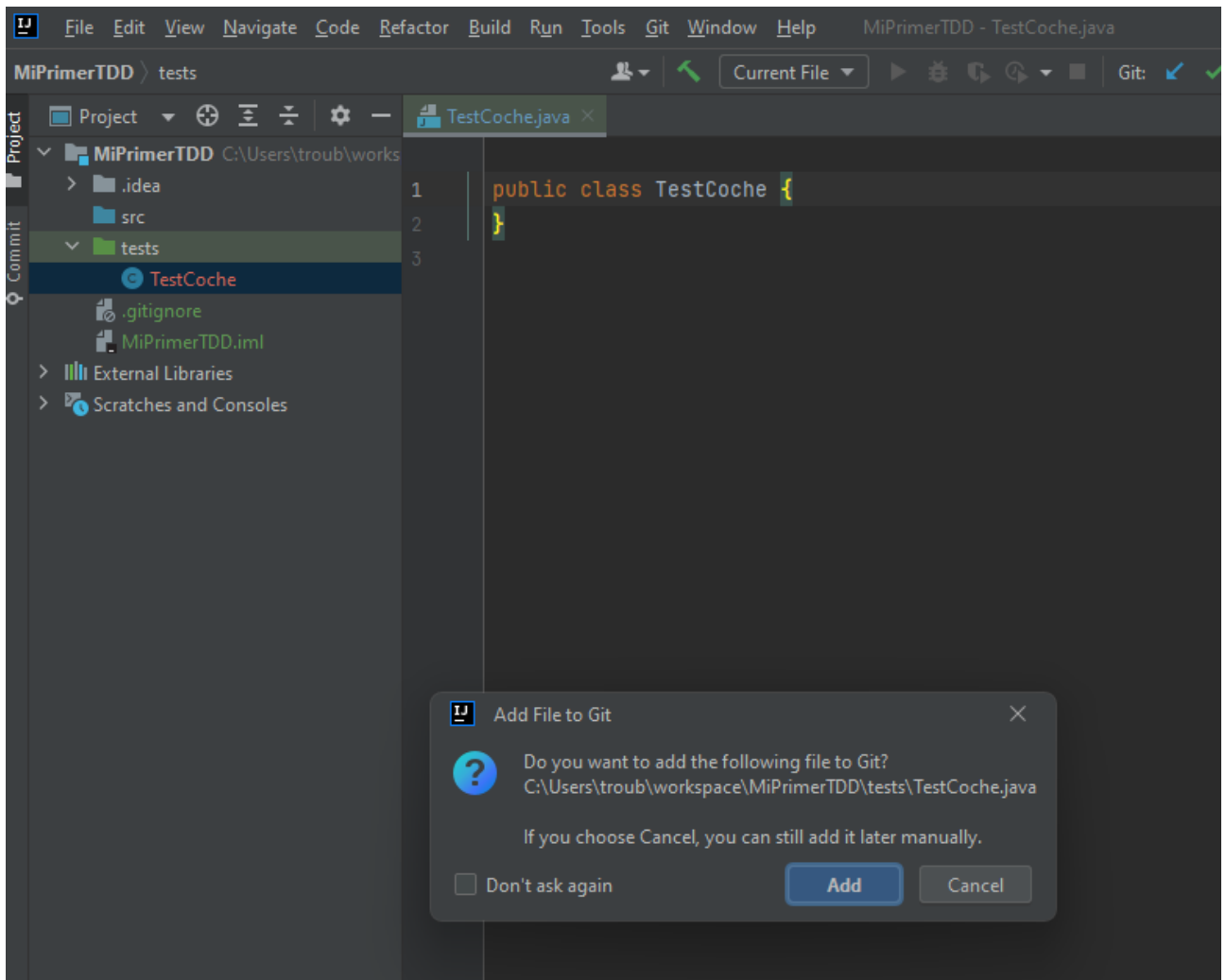


Luego hacemos click derecho sobre el nombre del proyecto, hacemos click en new y directorio.

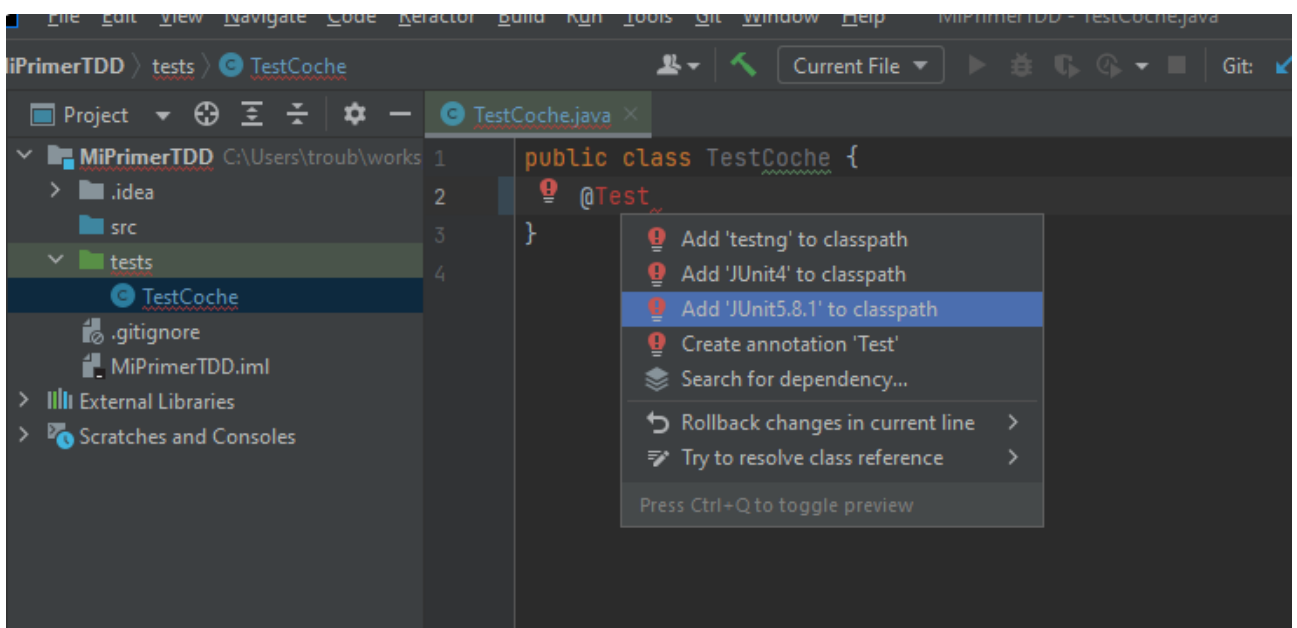


Ahora en la carpeta test hacemos click derecho → mark directory as → Source Test.
Ahora click derecho al mismo, y creamos una clase java.

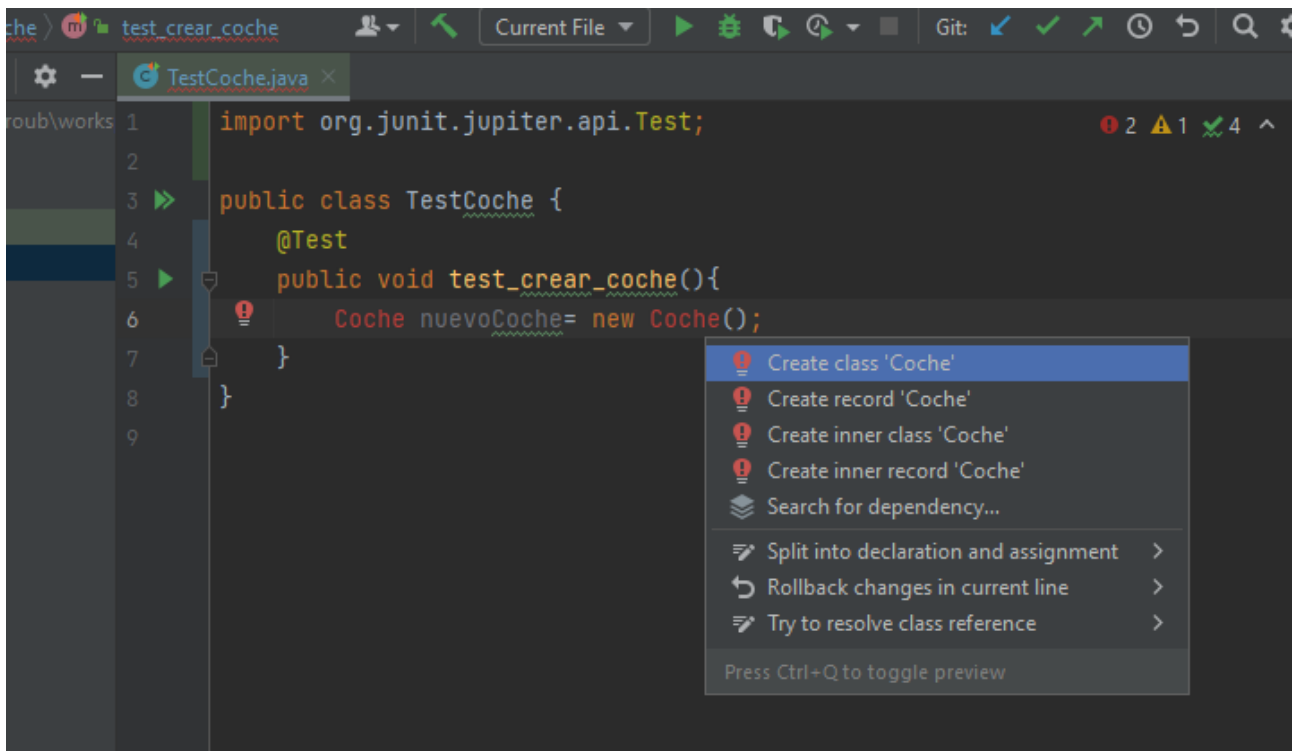




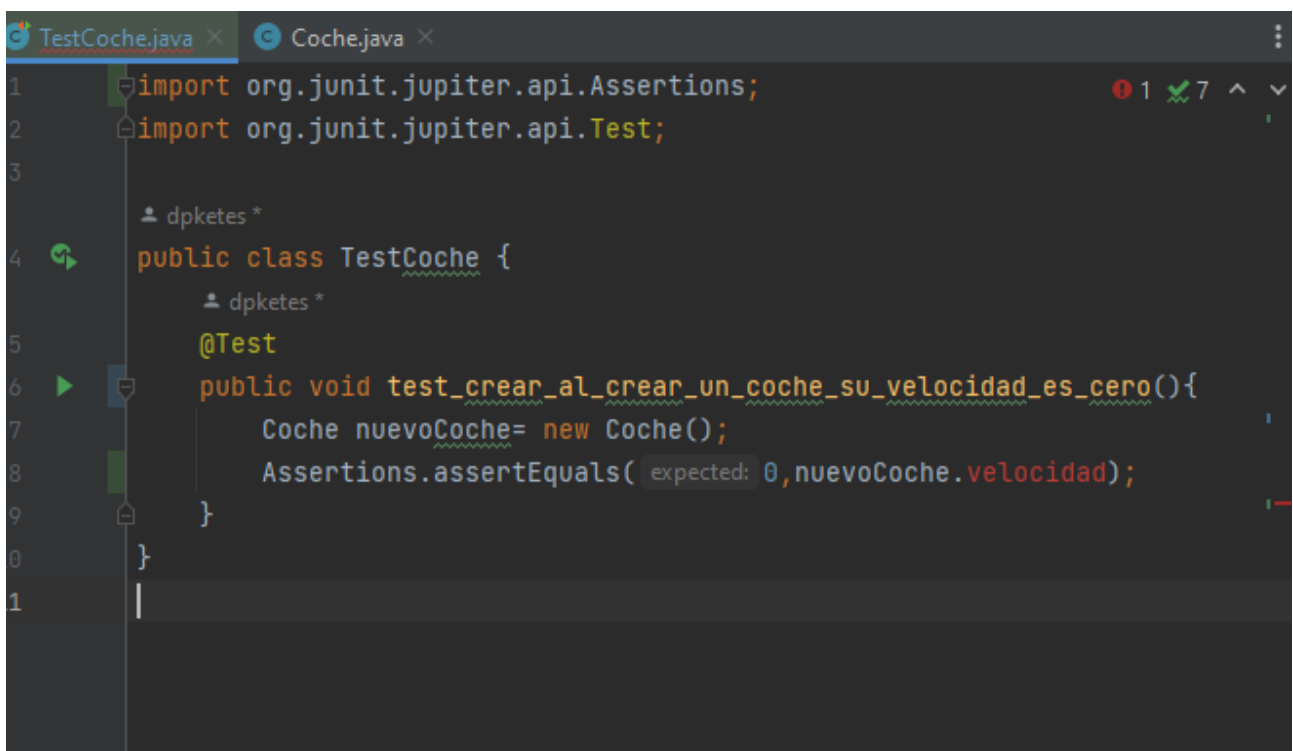
Ponemos @Test y clickamos click derecho para instalar el JUnit en el archivo.

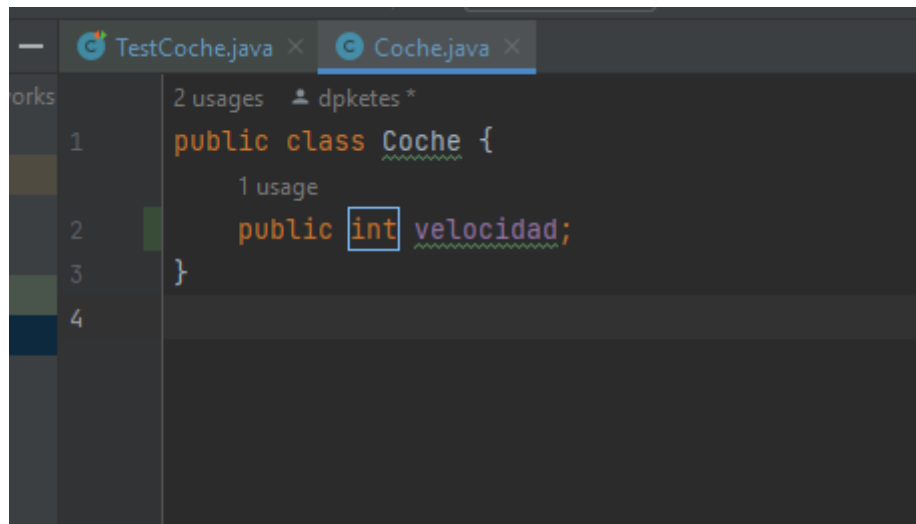


Creamos en el test un método el cual comprobara que se crea un objeto coche, para ello dentro de este método abra que crear el objeto coche el cual nos dara error ya que no existe ninguna clase coche, por lo tanto en el error click derecho y crear clase coche en el directorio src.



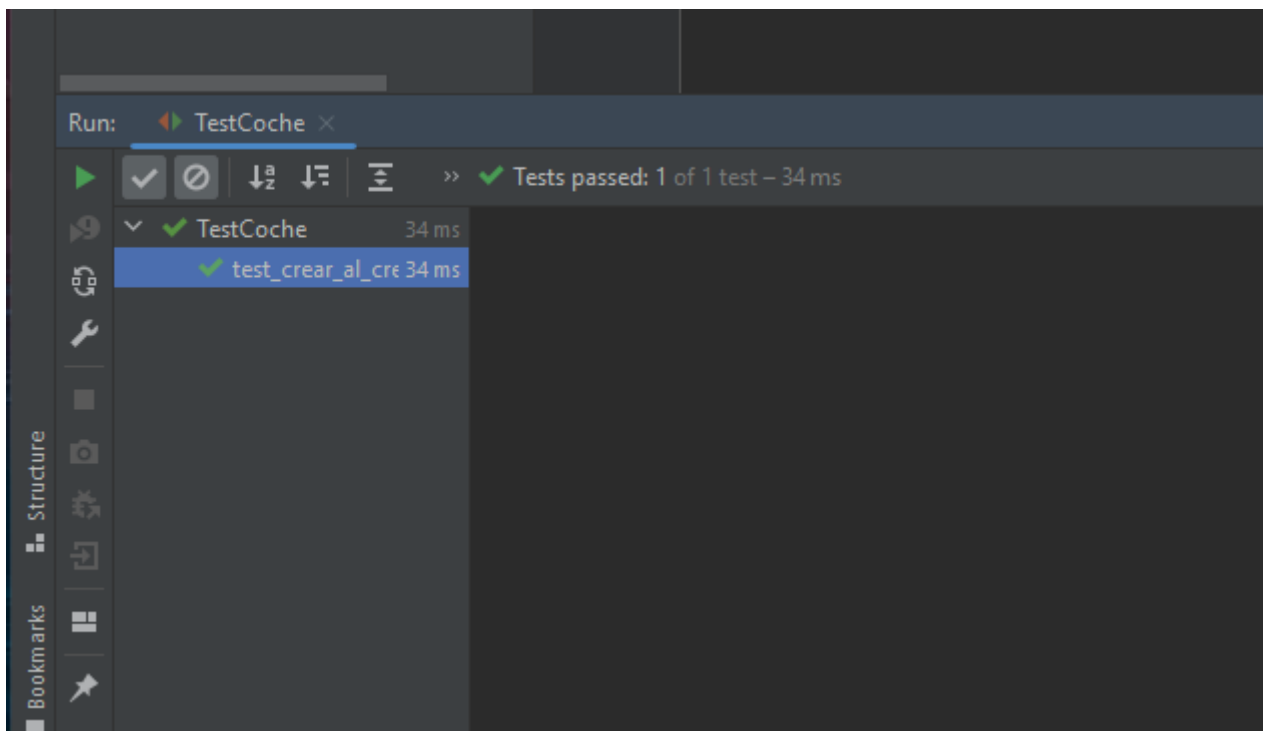
Cambiamos el anterior metodo, para que ahora compruebe que al crear un coche su velocidad es cero. Para ello tendremos que crear un atributo llamado velocidad.





2 usages dpketes *

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



Run: TestCoche x

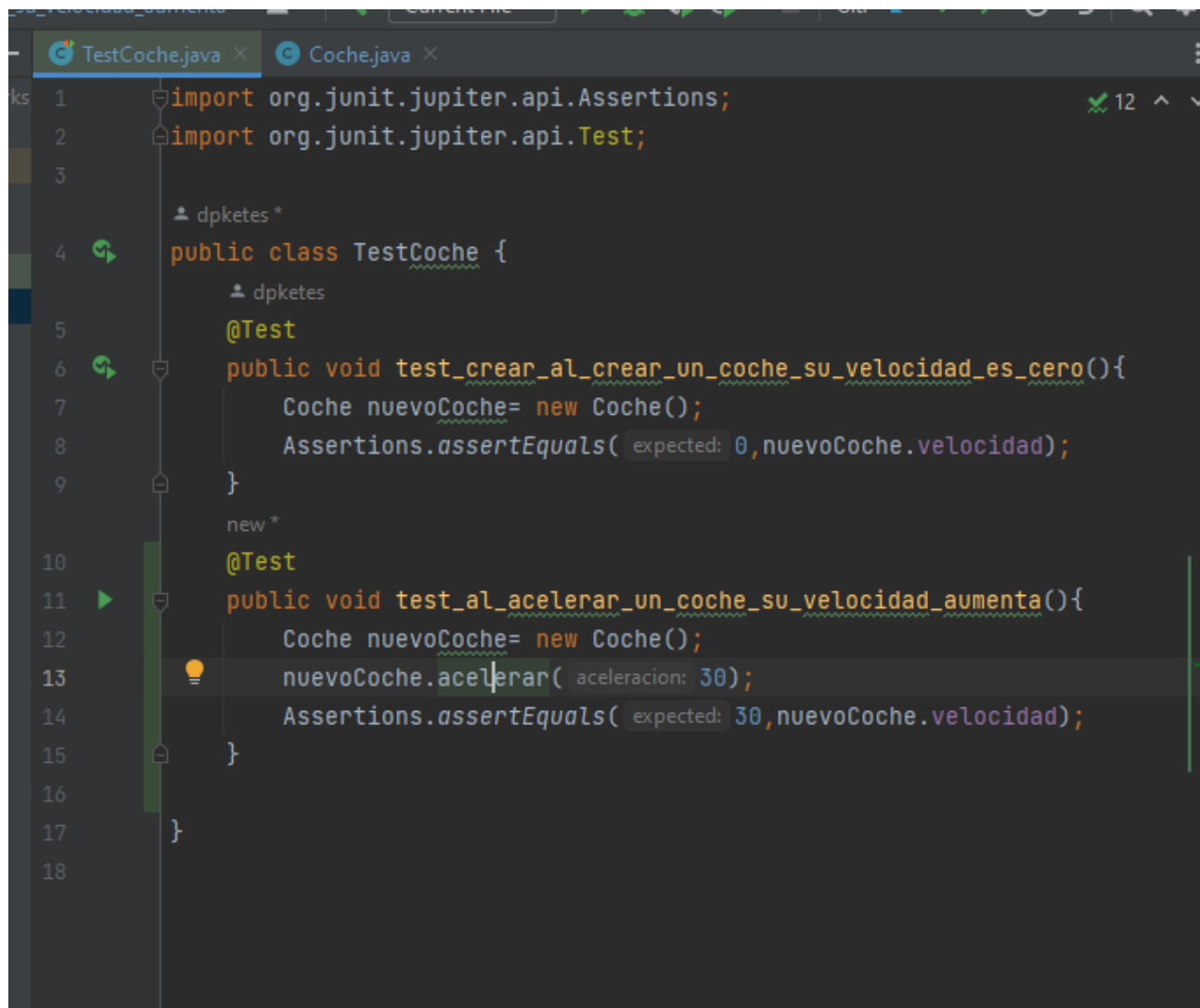
✓ Tests passed: 1 of 1 test – 34 ms

- ✓ TestCoche 34 ms
 - ✓ test_crear_al_cre 34 ms

Structure

Bookmarks

Ahora vamos a proceder a realizar los metodos acelerar, decelerar y que al decelerar la velocidad no sea menor que cero. Para ello en la clase test hacemos los mismos pasos que con el anterior metodo. Especificando las velocidades esperadas despues de acelerar y despues de decelerar, en sus respectivos metodos.



```
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_crear_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( aceleracion: 30);
15         Assertions.assertEquals( expected: 30,nuevoCoche.velocidad);
16     }
17 }
18
```

```
acelerar
b\works
TestCoche.java x Coche.java x
4 usages dpketes *
1 public class Coche {
2     3 usages
3     public int velocidad;
4     1 usage new *
5     public void acelerar(int aceleracion) {
6         velocidad += aceleracion;
7     }
8 }
```

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



```

1 usage new*
public void decelerar(int deceleracion) {
    velocidad -= deceleracion;
}

```

```

}
new*
@Test
public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_de_cero(){
    Coche nuevoCoche= new Coche();
    nuevoCoche.velocidad = 50;
    nuevoCoche.decelerar( deceleracion: 80);
    Assertions.assertEquals( expected: 0,nuevoCoche.velocidad);
}
}

```

Aqui hacemos comprobación antes de definir en el metodo decelerar de la clase coche que velocidad no puede ser menor que cero.

Run: TestCoche x

Tests failed: 1, passed: 3 of 4 tests – 36 ms

TestCoche 36 ms

- test_crear_al_crear 31 ms
- test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_de_cero 4 ms**
- test_al_acelerar_un_coche 1 ms
- test_al_decelerar_un_coche 1 ms

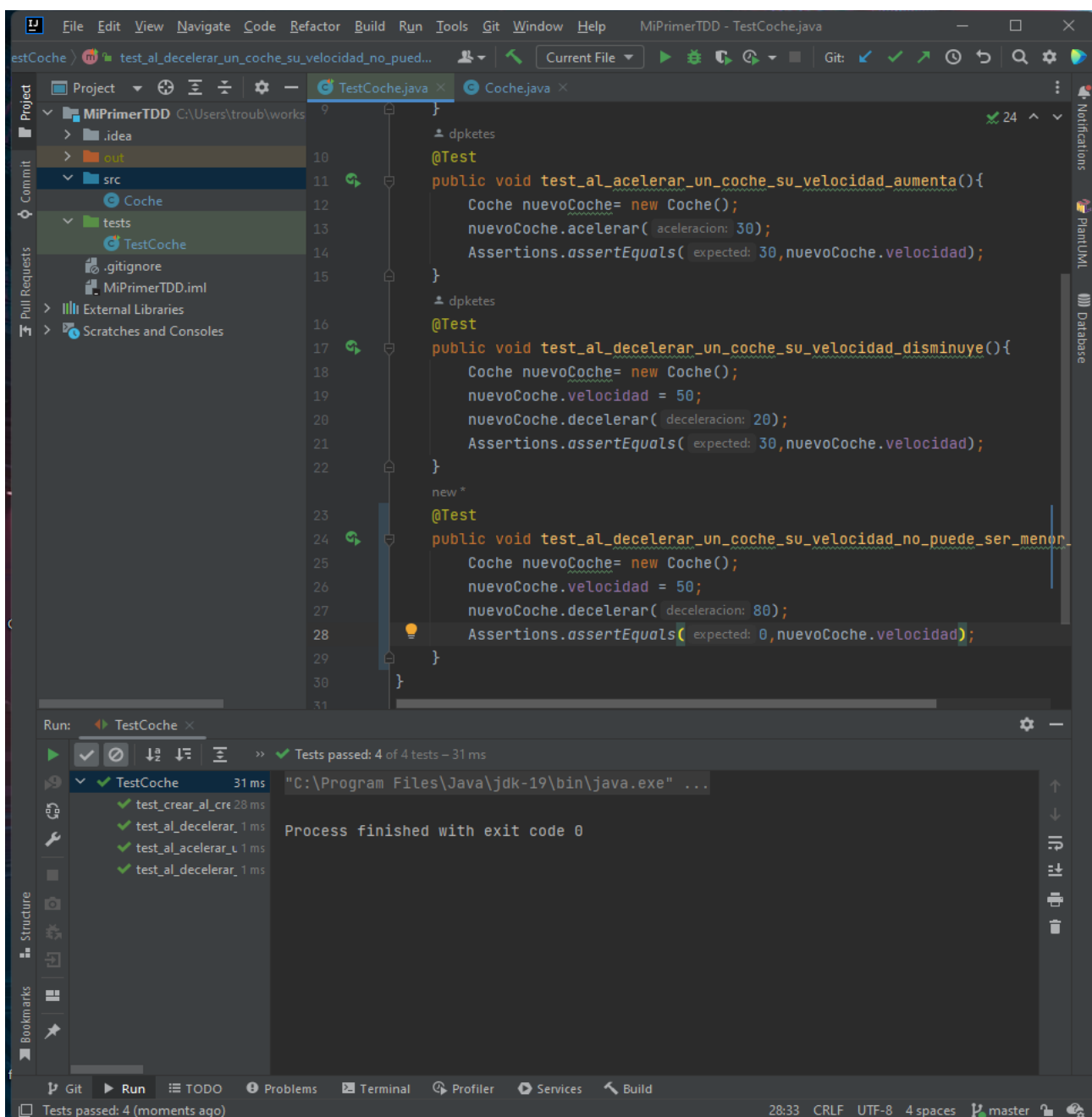
org.opentest4j.AssertionFailedError:
Expected :0
Actual :-30
[Click to see difference](#)

<5 internal lines>

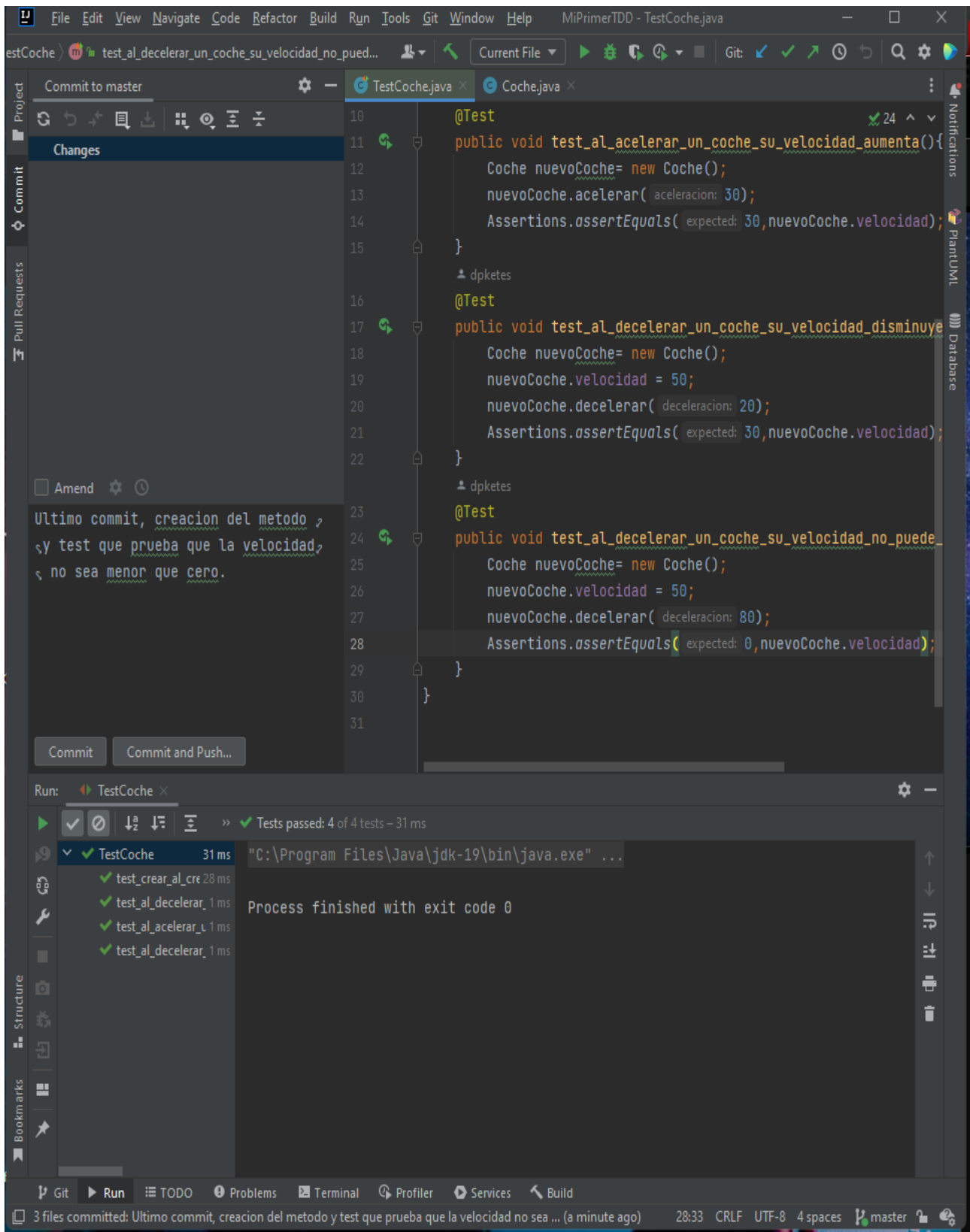
- at TestCoche.test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_de_cero
- at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
- at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <27 internal lines>

Definimos en el metodo decelerar de la clase coche que velocidad no puede ser menor que cero.

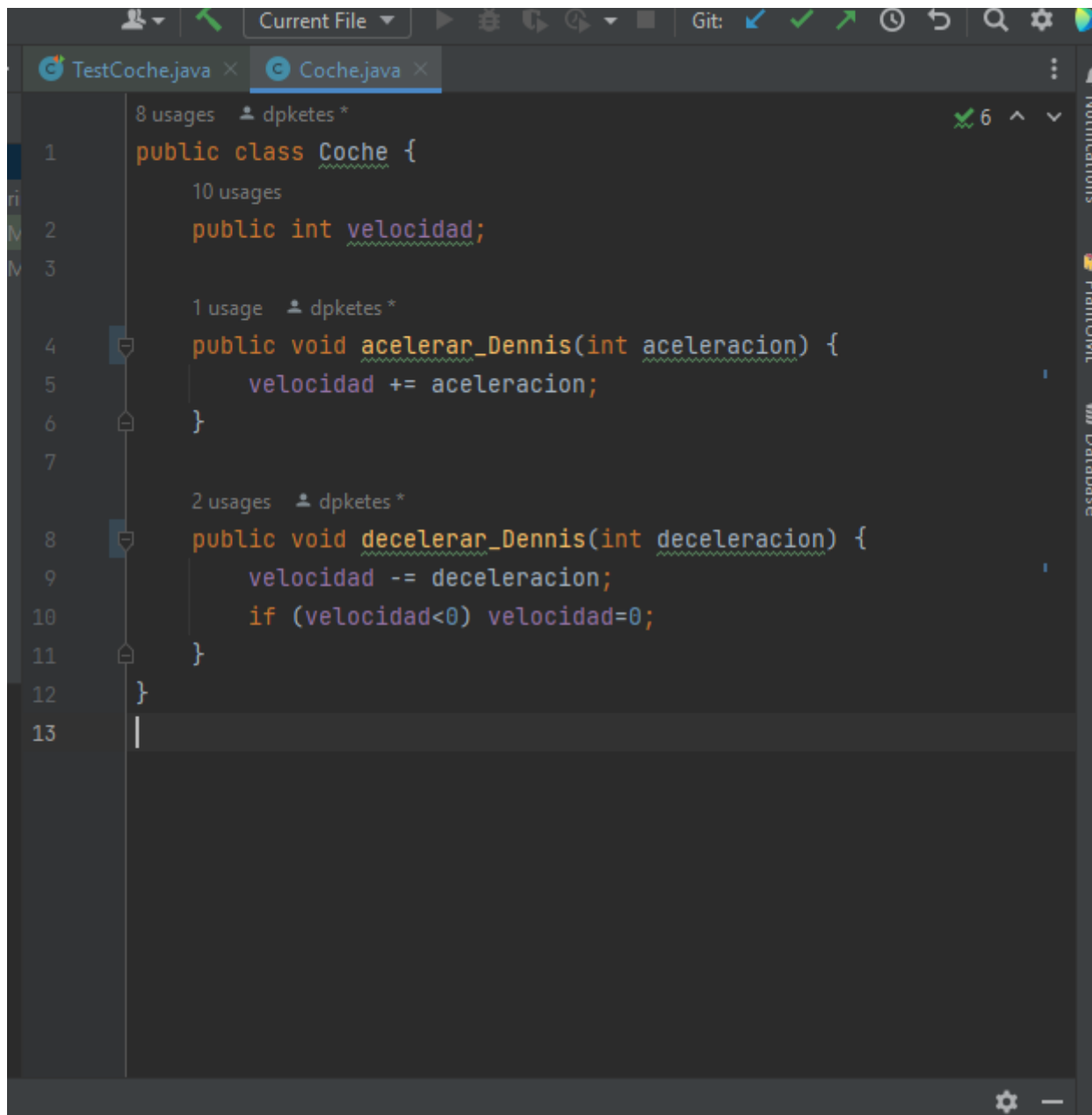
```
2 usages  dpketes *
public void decelerar(int deceleracion) {
    velocidad -= deceleracion;
    if (velocidad < 0) velocidad = 0;
}
```



Después de terminar con todos los commits y guardados, procedemos a crear la rama Refactorizado, y cambiamos todos los metodos por metodo+mi_nombre.



```
TestCoche.java x Coche.java x
4
5
6 ▶ Crear_un_coche_su_velocidad_es_cero_Dennis(){
7     coche();
8     : expected: 0,nuevoCoche.velocidad);
9
10
11 ▶ Crear_un_coche_su_velocidad_aumenta_Dennis(){
12     coche();
13     aceleracion: 30);
14     : expected: 30,nuevoCoche.velocidad);
15
16
17 ▶ Crear_un_coche_su_velocidad_disminuye_Dennis(){
18     coche();
19     i0;
20     aceleracion: 20);
21     : expected: 30,nuevoCoche.velocidad);
22
23
24 ▶ Crear_un_coche_su_velocidad_no_puede_ser_menor_de_cero_Dennis(){
25     coche();
```



The screenshot shows an IDE window with two tabs: 'TestCoche.java' and 'Coche.java'. The 'Coche.java' tab is active, displaying the following Java code:

```
1 public class Coche {  
2     public int velocidad;  
3  
4     public void acelerar_Dennis(int aceleracion) {  
5         velocidad += aceleracion;  
6     }  
7  
8     public void decelerar_Dennis(int deceleracion) {  
9         velocidad -= deceleracion;  
10        if (velocidad<0) velocidad=0;  
11    }  
12 }  
13 |
```

Annotations in the IDE indicate usage counts: '8 usages' for the class, '10 usages' for the 'velocidad' attribute, '1 usage' for the 'acelerar_Dennis' method, and '2 usages' for the 'decelerar_Dennis' method. The code is color-coded with orange for keywords, green for comments, and blue for identifiers. The IDE interface includes a top toolbar with icons for file operations, a search bar, and a Git status bar at the bottom right.

Y esto seria todo. He realizado unos 6 commits en la rama master, y un commit en la rama Refactorizado.