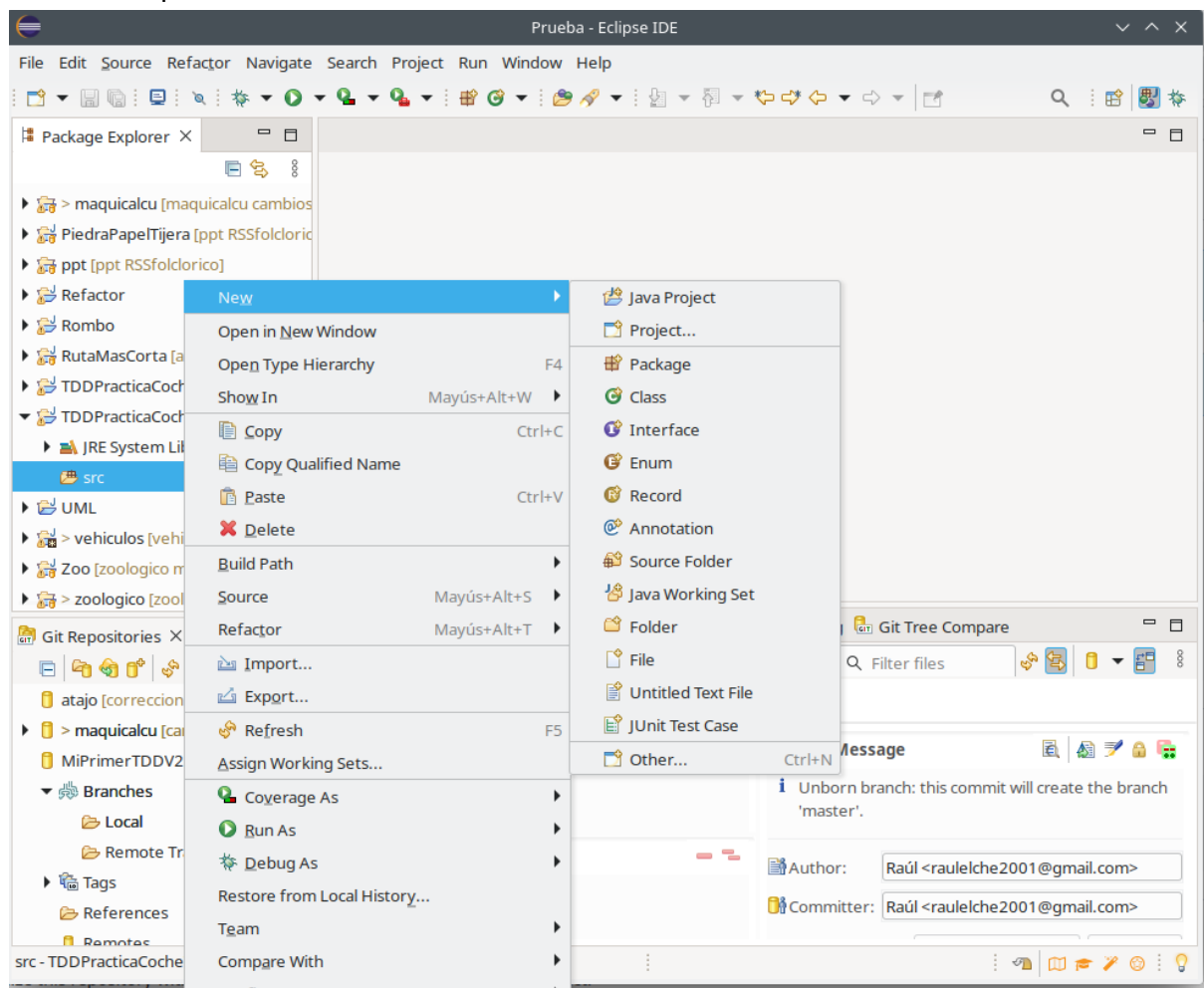
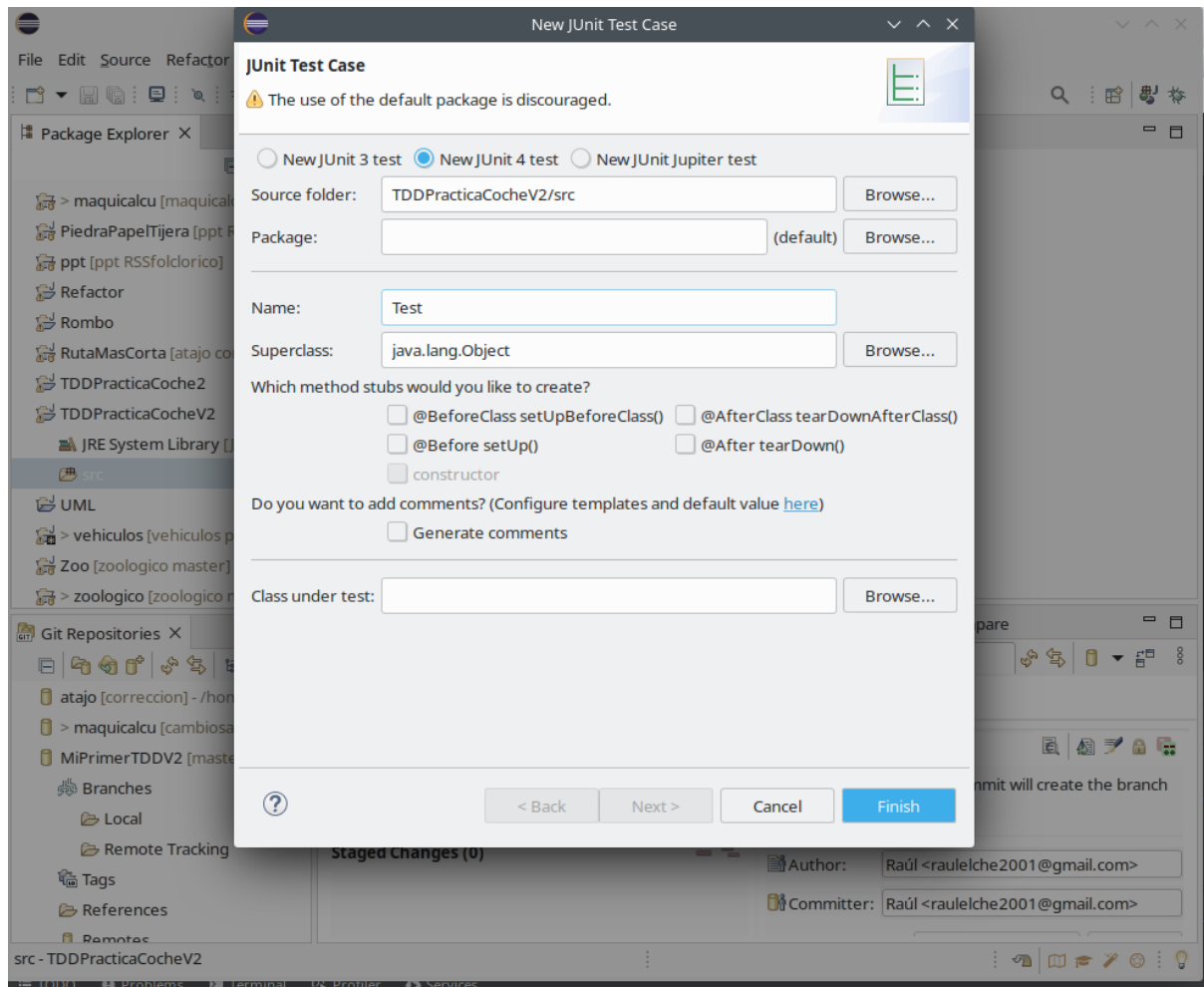


ENTORNOS: Mi primer TDD V2

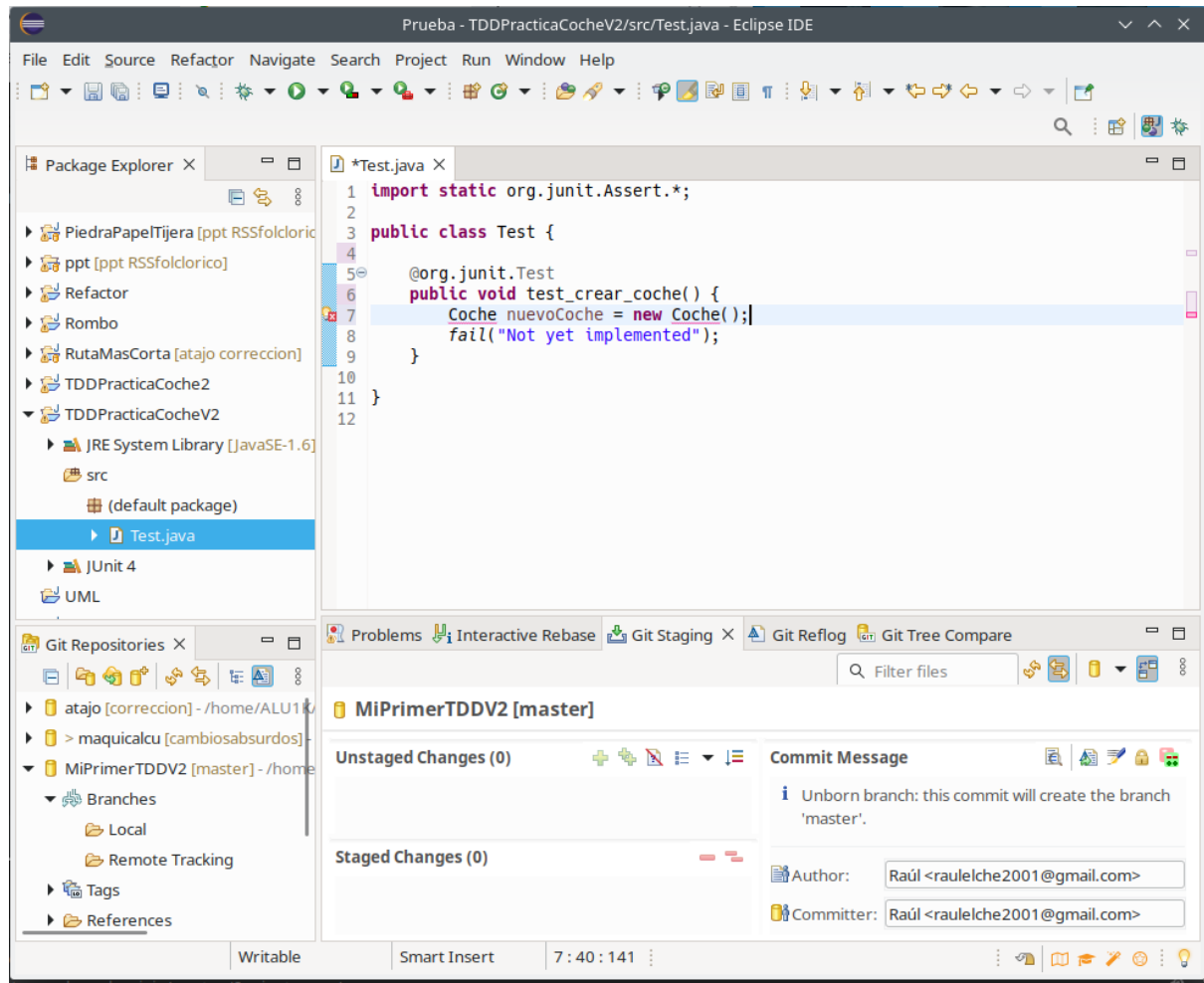
Antes de comenzar con esta práctica, crearemos un proyecto. En la carpeta src, hacemos click derecho y creamos un “JUnit Test Case”, también podemos crear una carpeta test para incluirlo ahí pero no es estrictamente necesario.



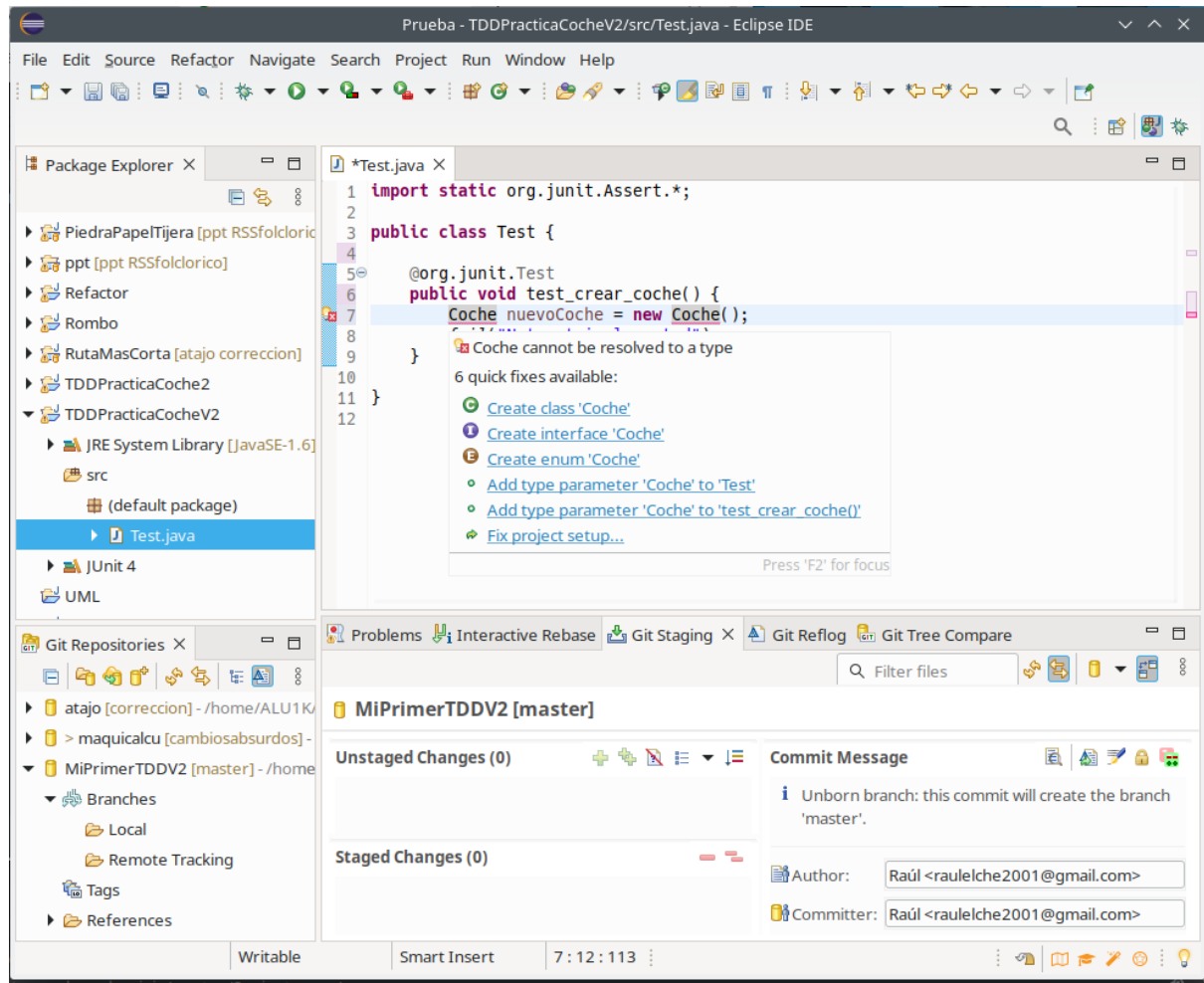
Marcamos la versión de JUnit que vamos a usar, en este caso hemos marcado JUnit4, y escribimos un nombre nosotros hemos elegido "Test".



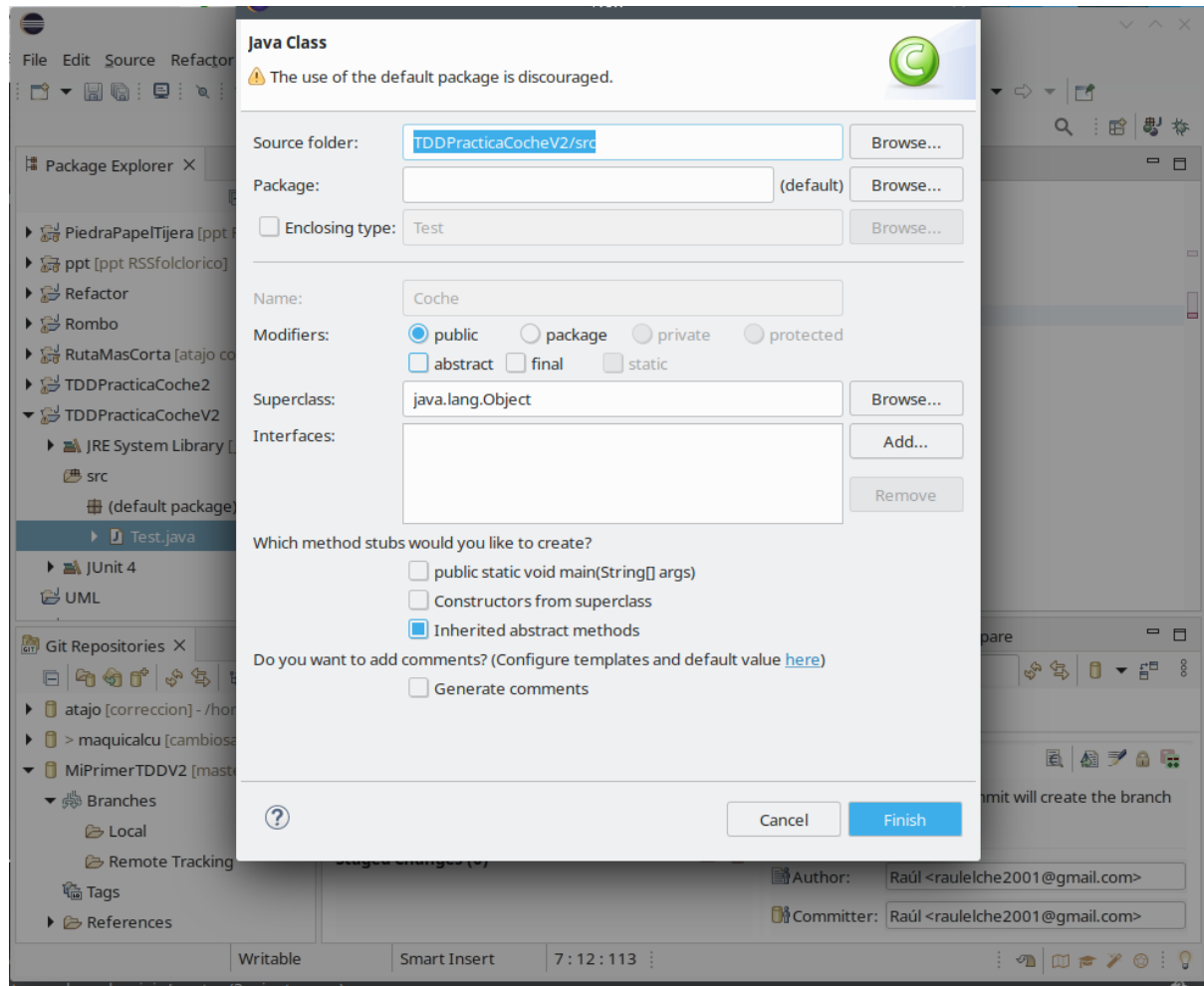
Se nos creará un test automático (no hice captura) y a partir de este crearemos el nuestro, cambiaremos el nombre del test predeterminado al que aparece en la captura y añadiremos la creación de un objeto Coche.



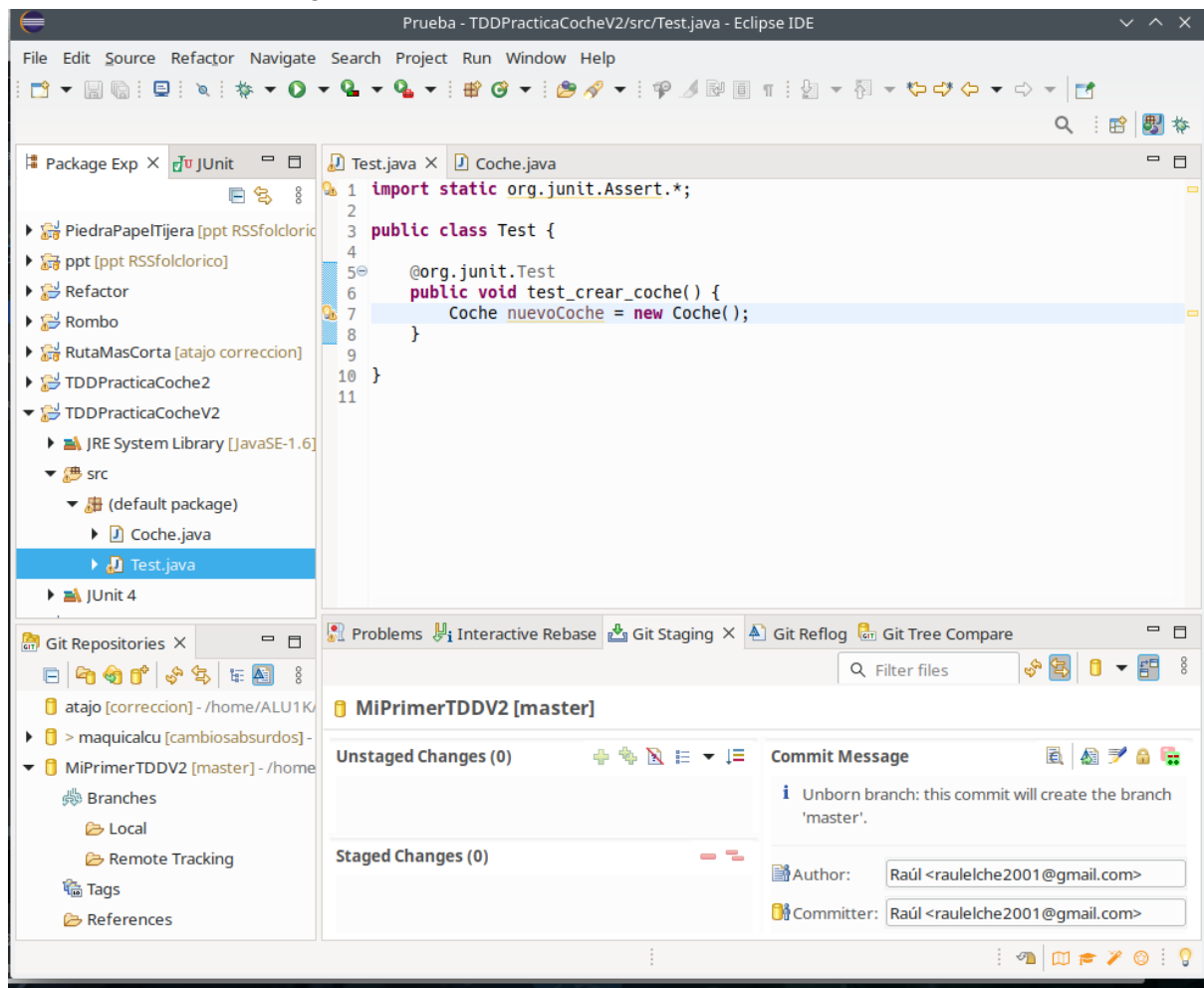
El objeto Coche aparece remarcado en rojo ya que no existe dicho objeto, por lo que clicamos en él y colocamos el puntero encima, nos sugerirá crear una clase, así que clicamos en “Create class ‘Coche’”.



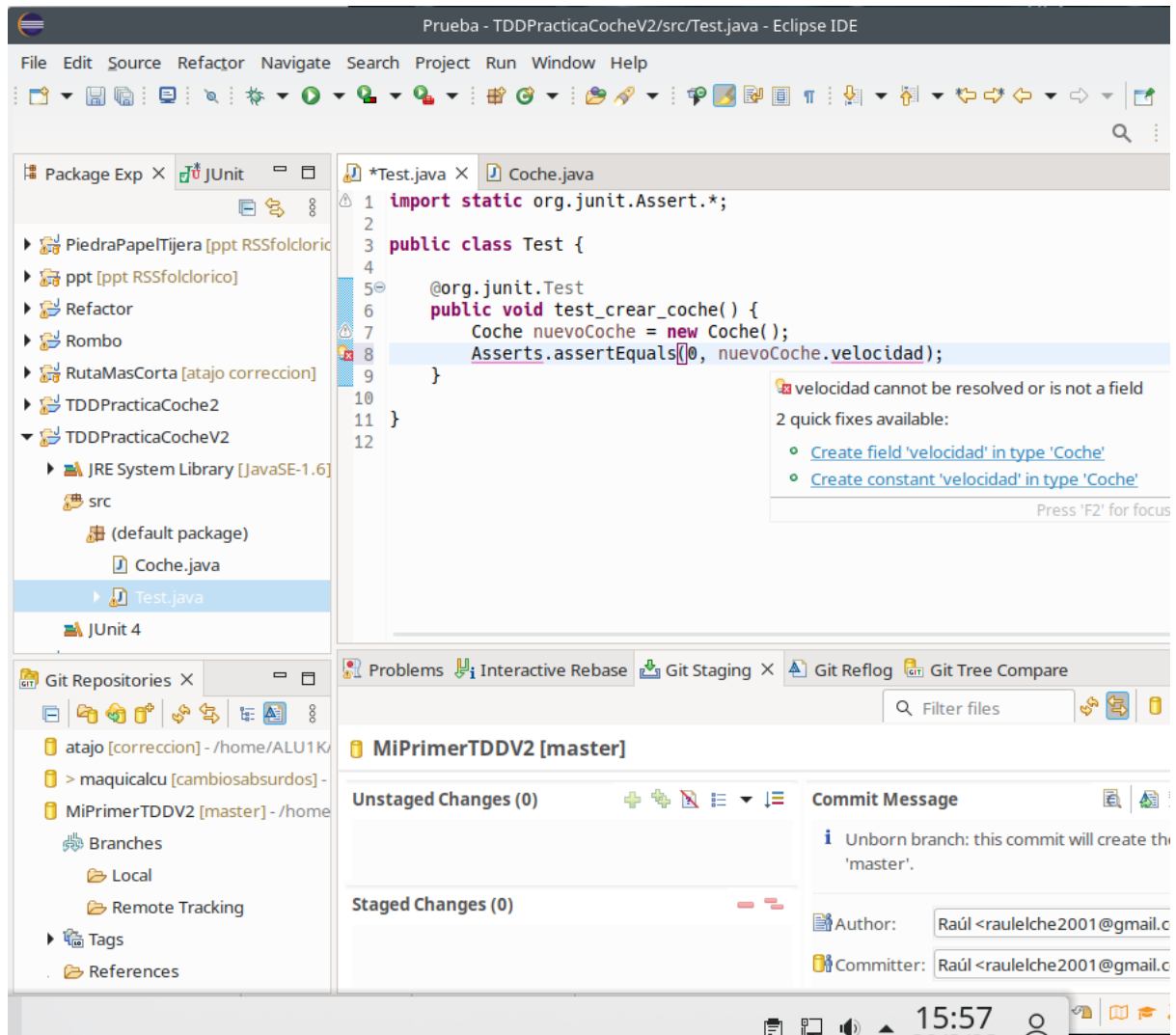
Clicamos en “Finish”.



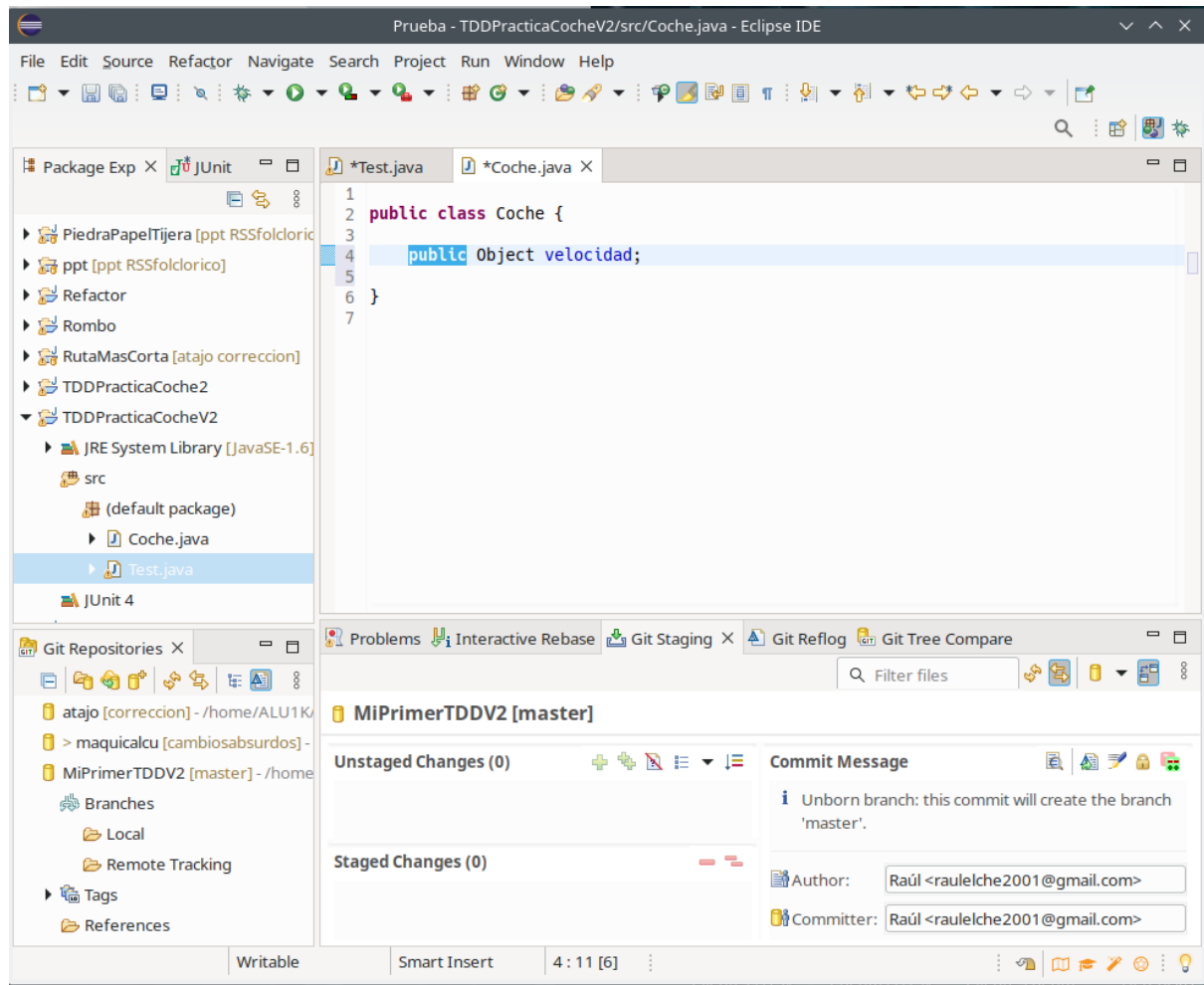
Y nos creará el objeto coche sin ningún atributo, constructor o método. Eliminamos el método fail que nos ha generado por defecto.



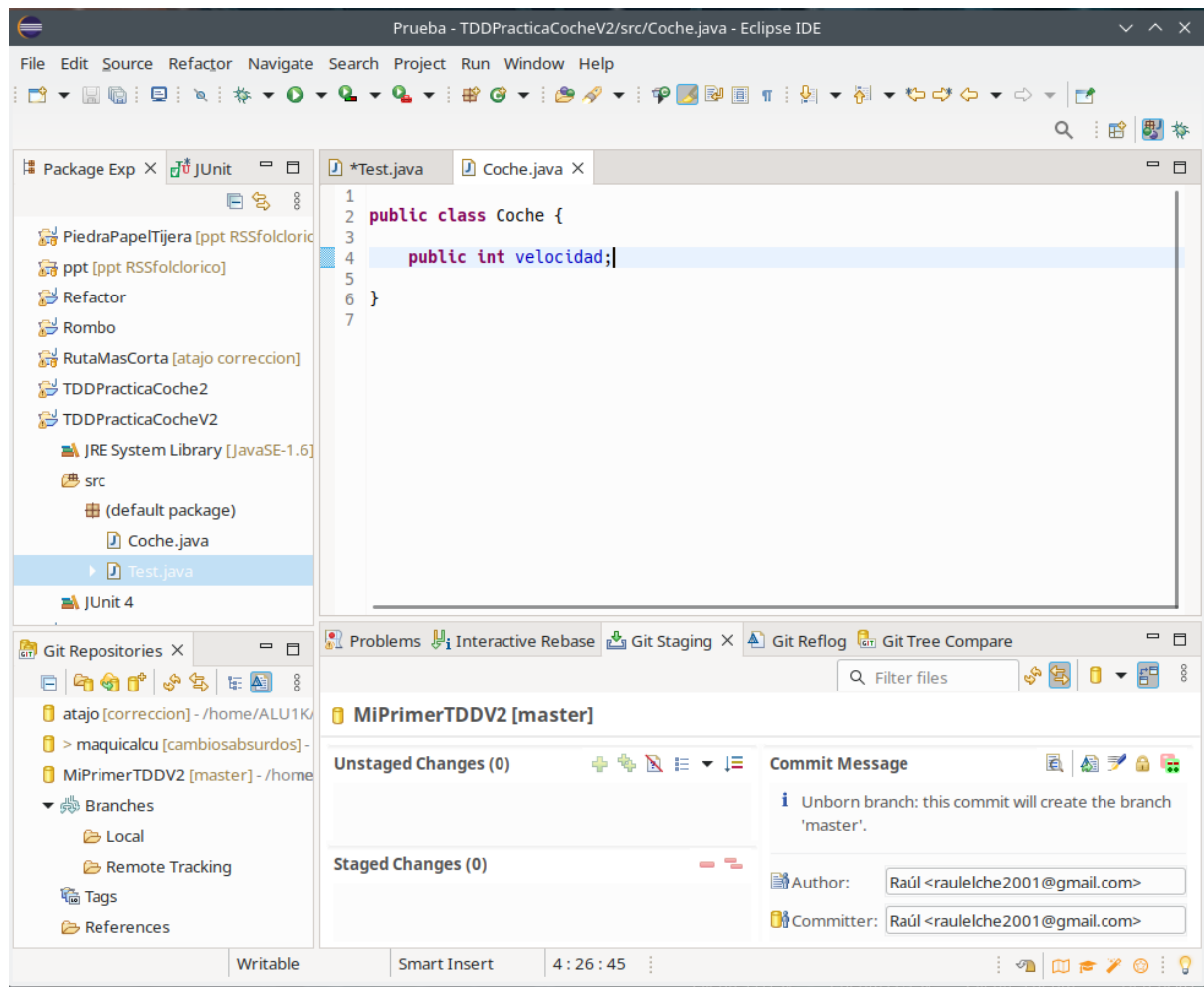
Creamos un método `Assert.equals()` en el cual se va a comparar que el coche que acabamos de crear tenga una velocidad de 0. El atributo `velocidad` del nuevo objeto `Coche` aparece en rojo debido a que este no existe, así que si clicamos en él y nos colocamos encima nos sugerirá crearlo, clicamos en “Create field ‘velocidad’ in type ‘Coche’”. (Asserts aparece en rojo porque sobra la s)



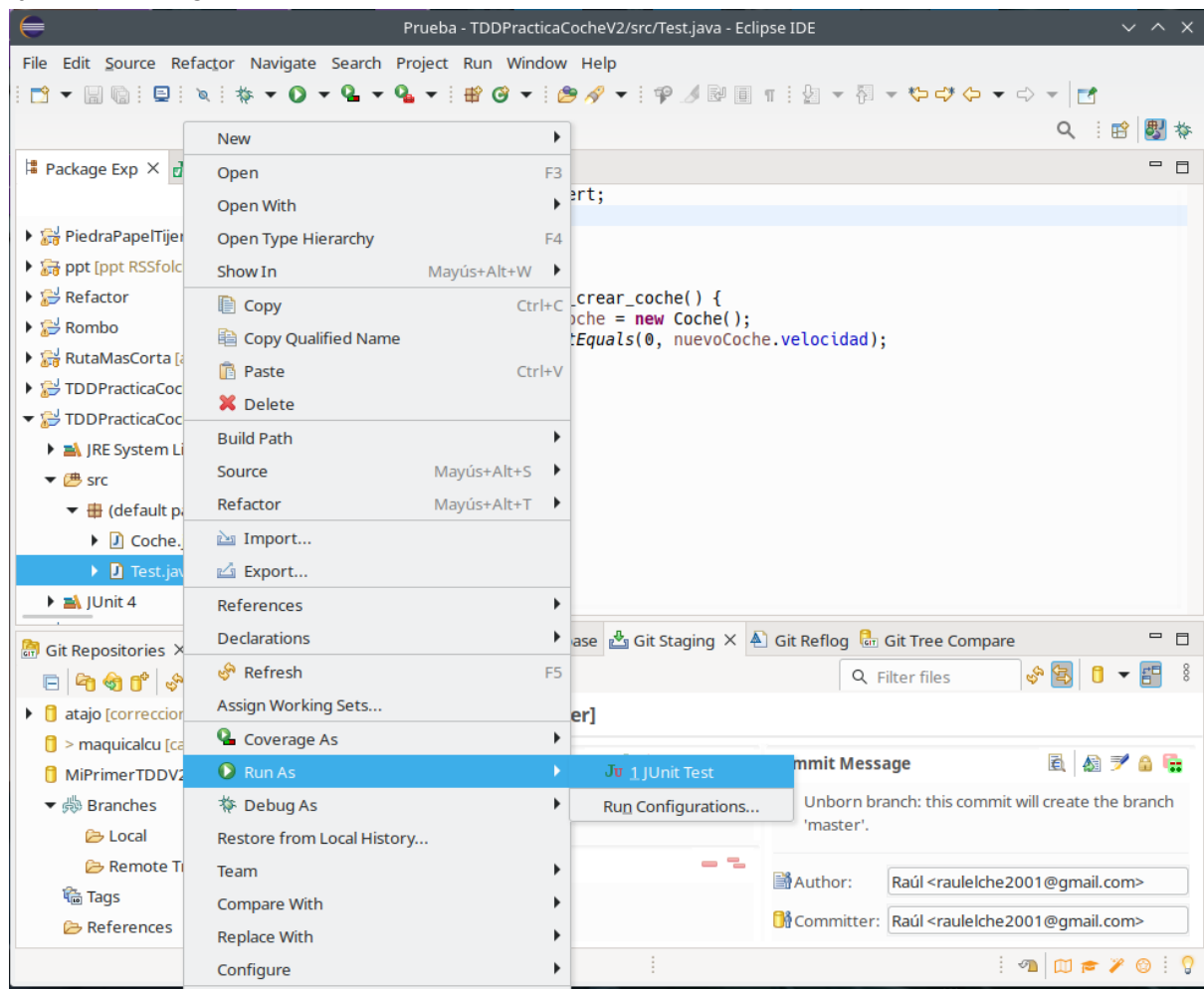
Nos lo creará por defecto y habrá que modificarlo, no será de tipo Object sino de tipo int.



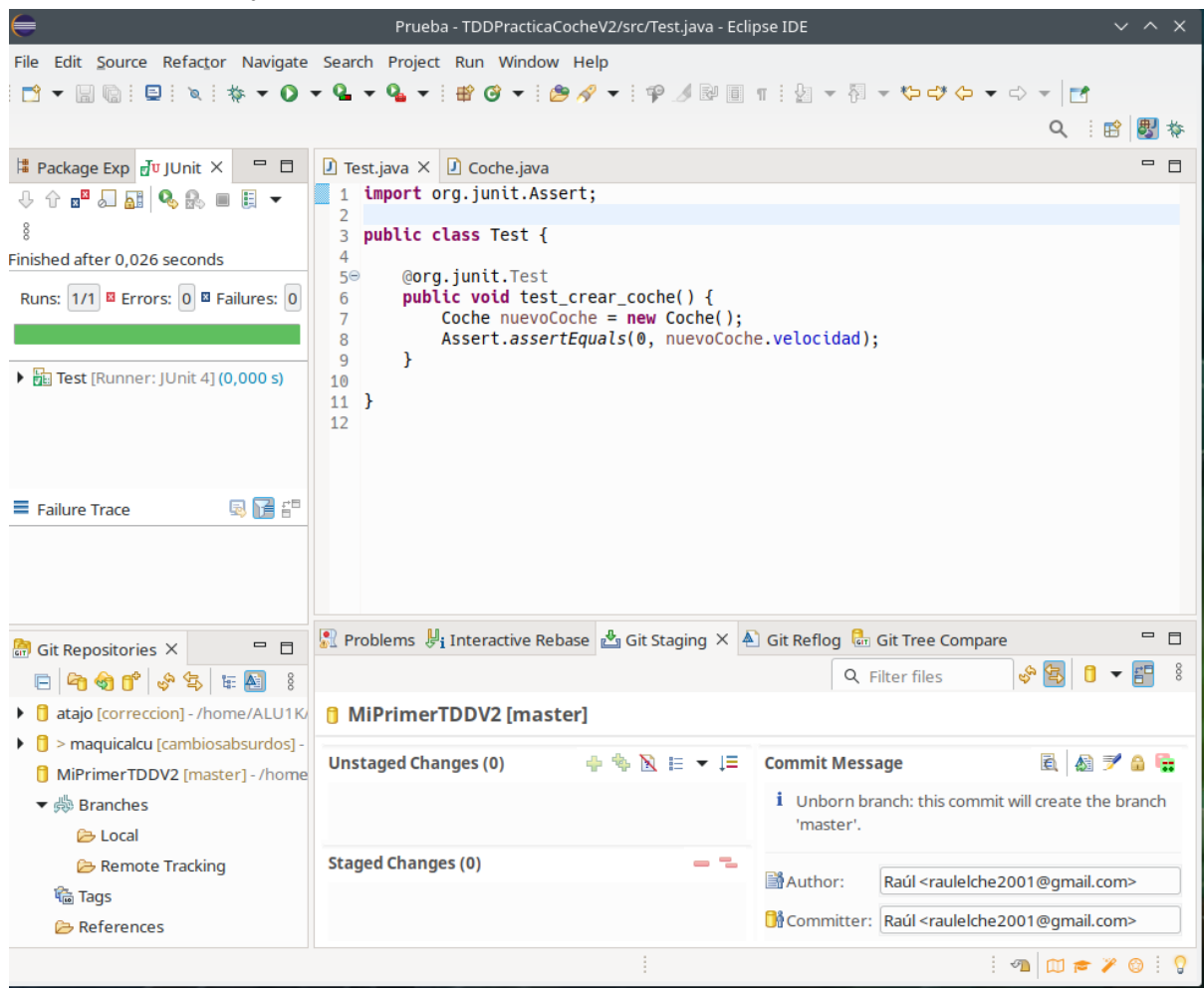
Quedaría de la siguiente forma.



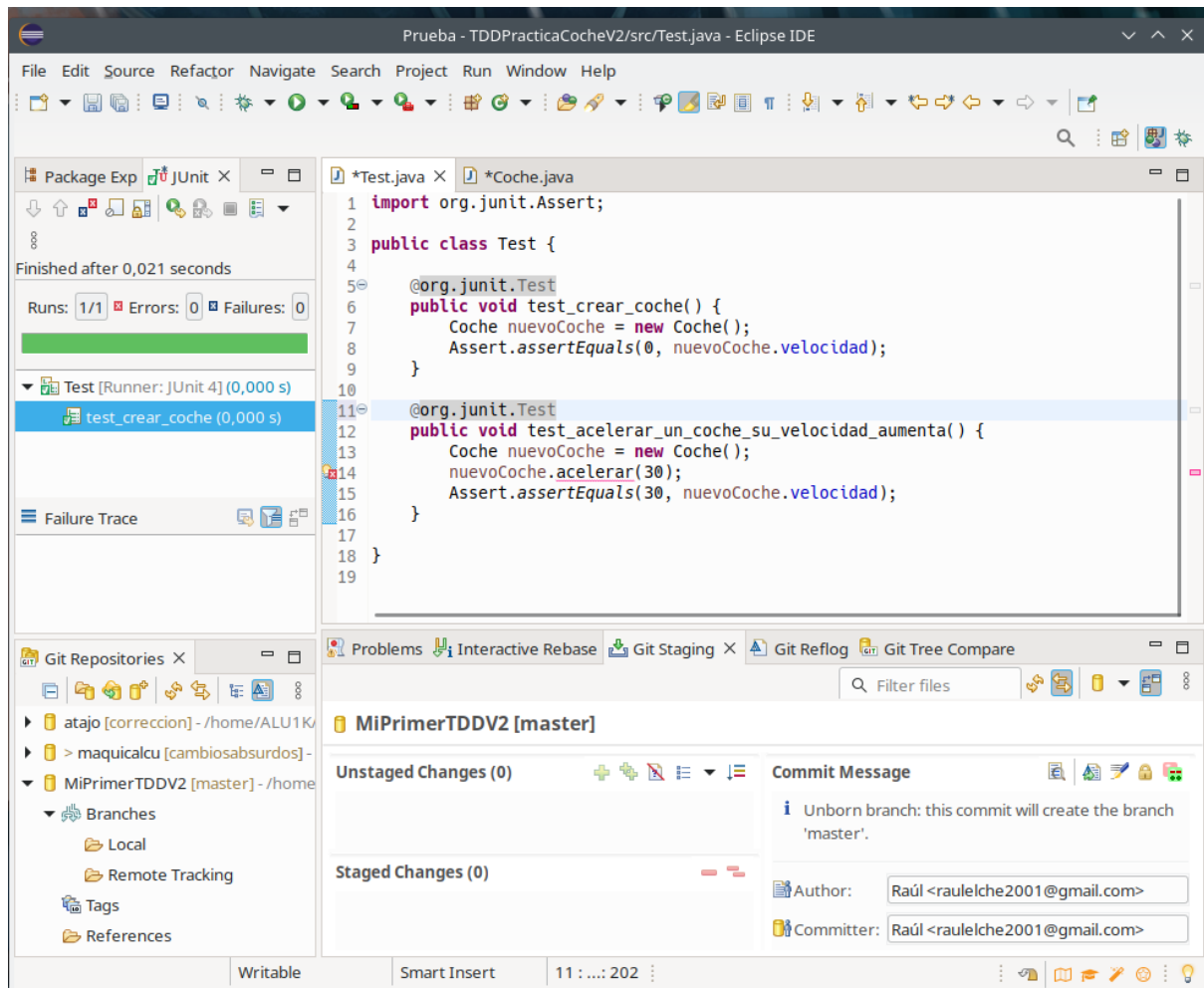
Ahora haciendo click derecho sobre Test.java y desplegando la opción “Run as” podemos ejecutar el programa como con un test clicando en “JUnit Test”.



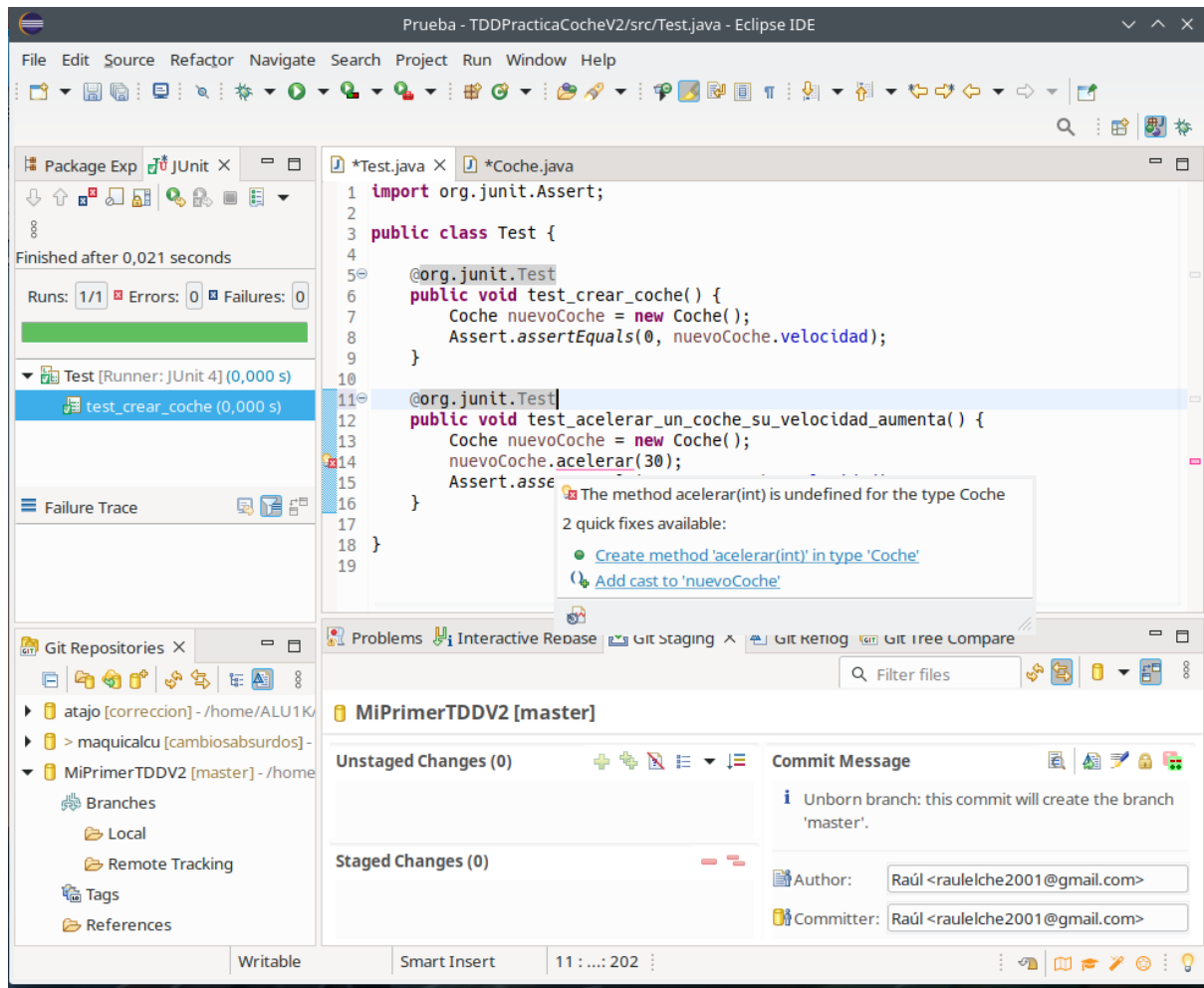
Y como podemos ver la pasamos sin errores debido a que la velocidad es 0 porque no la hemos modificado y un atributo int sin inicializar por defecto es 0.



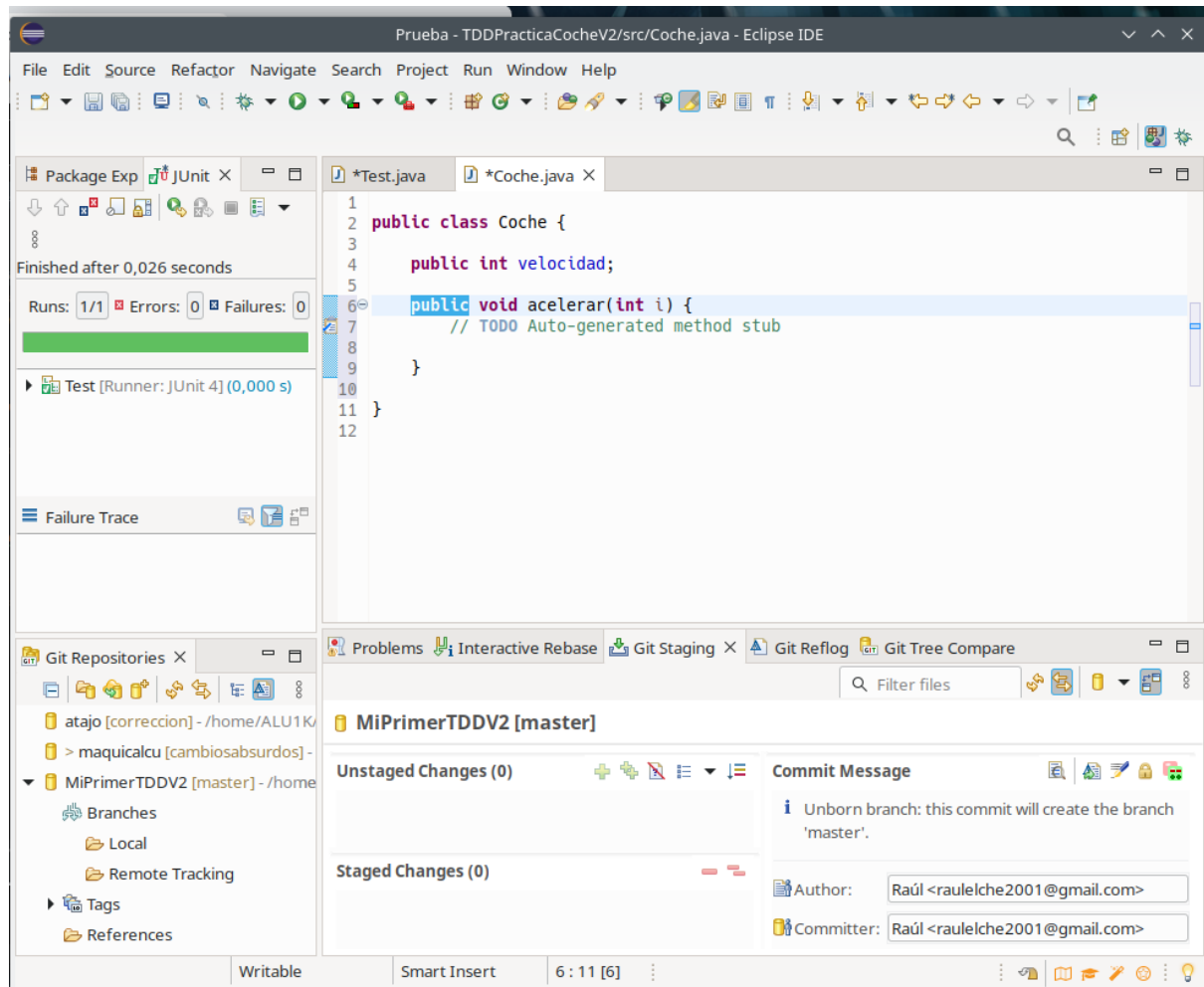
Ahora creamos otro test, en el que creamos un coche y este llama a un método de su clase que se llama acelerar y le pasará por parámetro un entero, en este caso 30. Además, el método `Assert.equals()` comparará la velocidad con 30, ya que queremos que el método acelerar aumente la velocidad del coche según lo que le pasemos por parámetro.



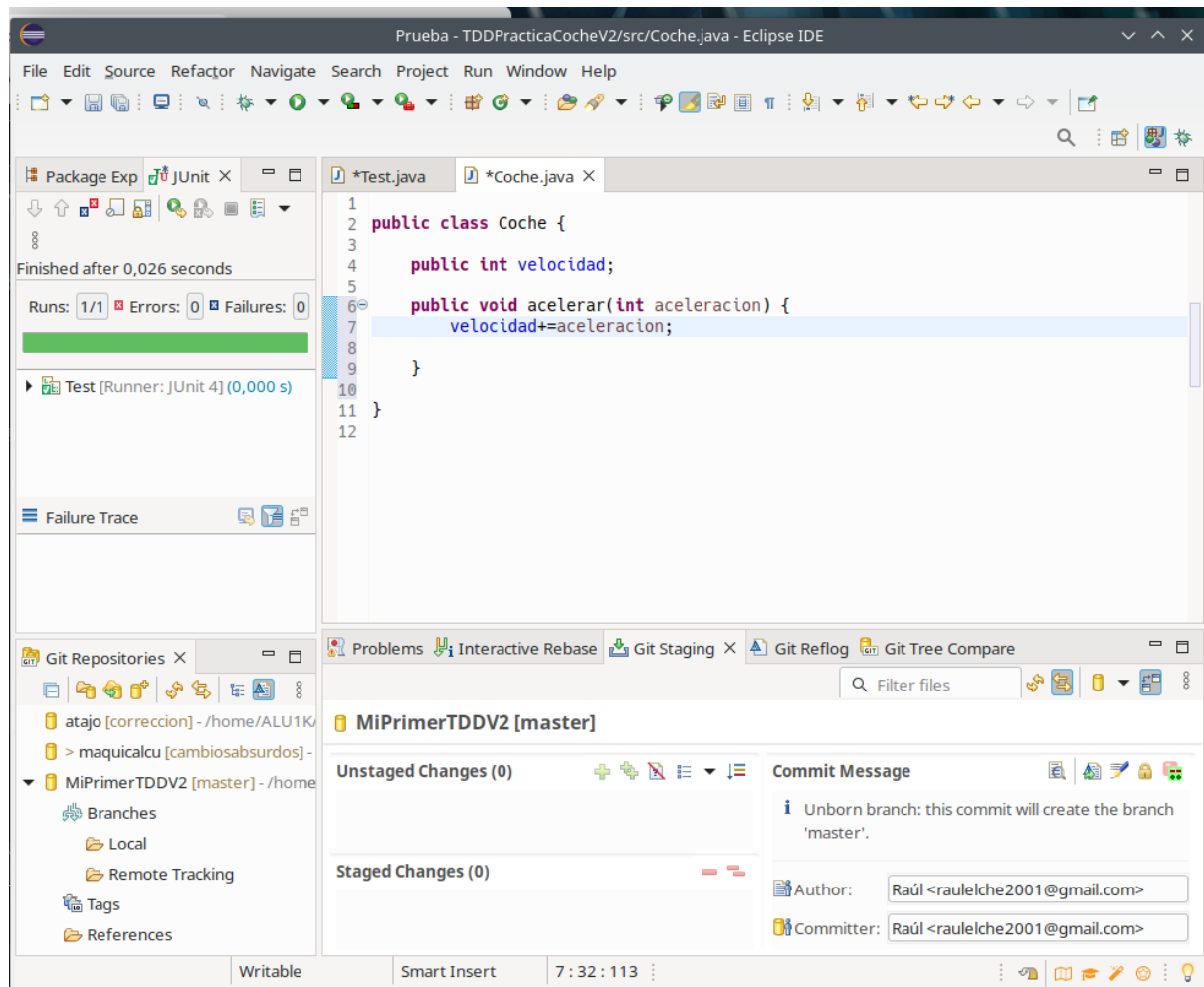
Ahora bien, como el método acelerar no existe aparecerá en rojo, clicamos y nos colocamos en él y seleccionamos “Create method ‘acelerar’ in type ‘Coche’”.



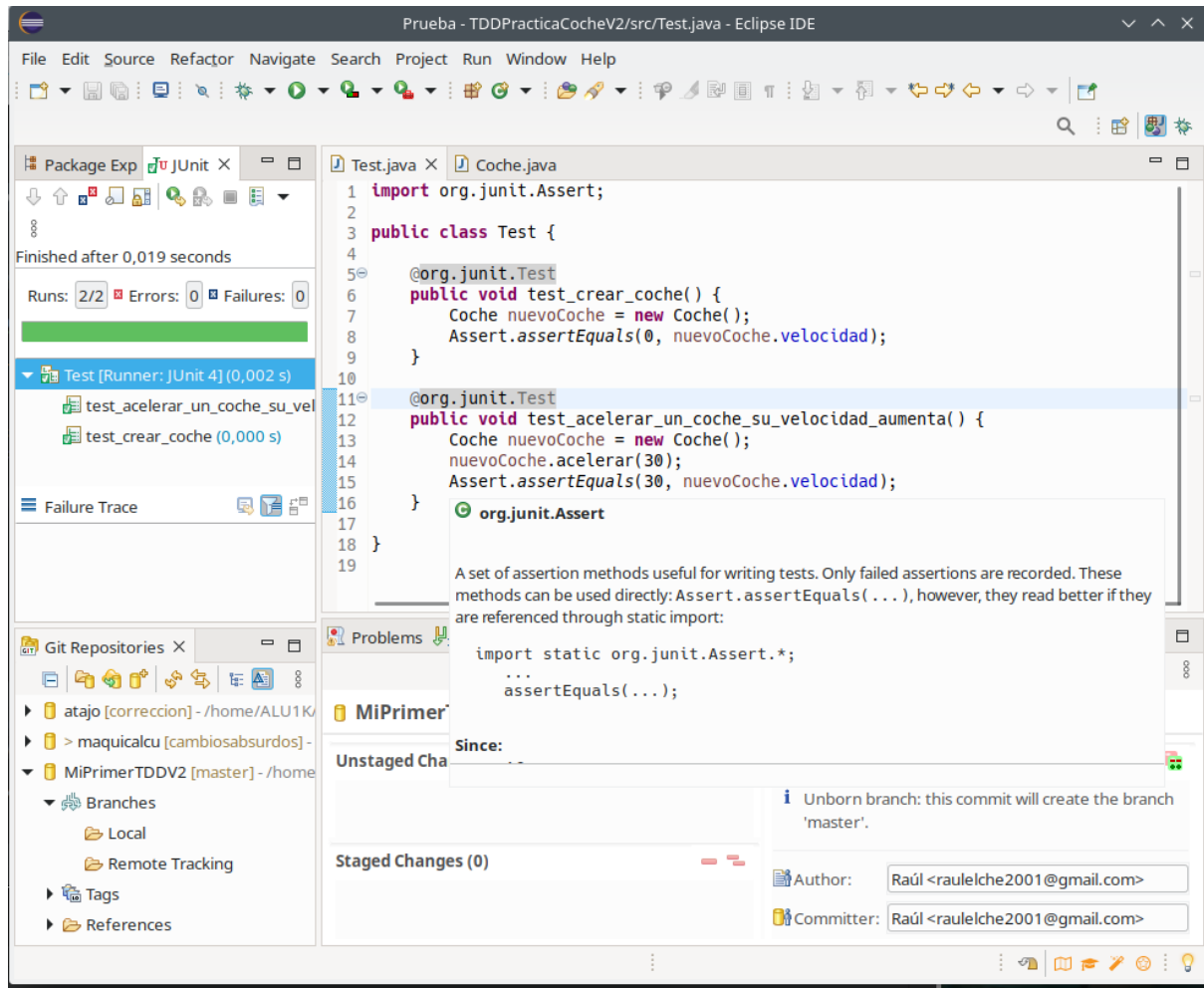
Crearé un método por defecto que tendremos que modificar.



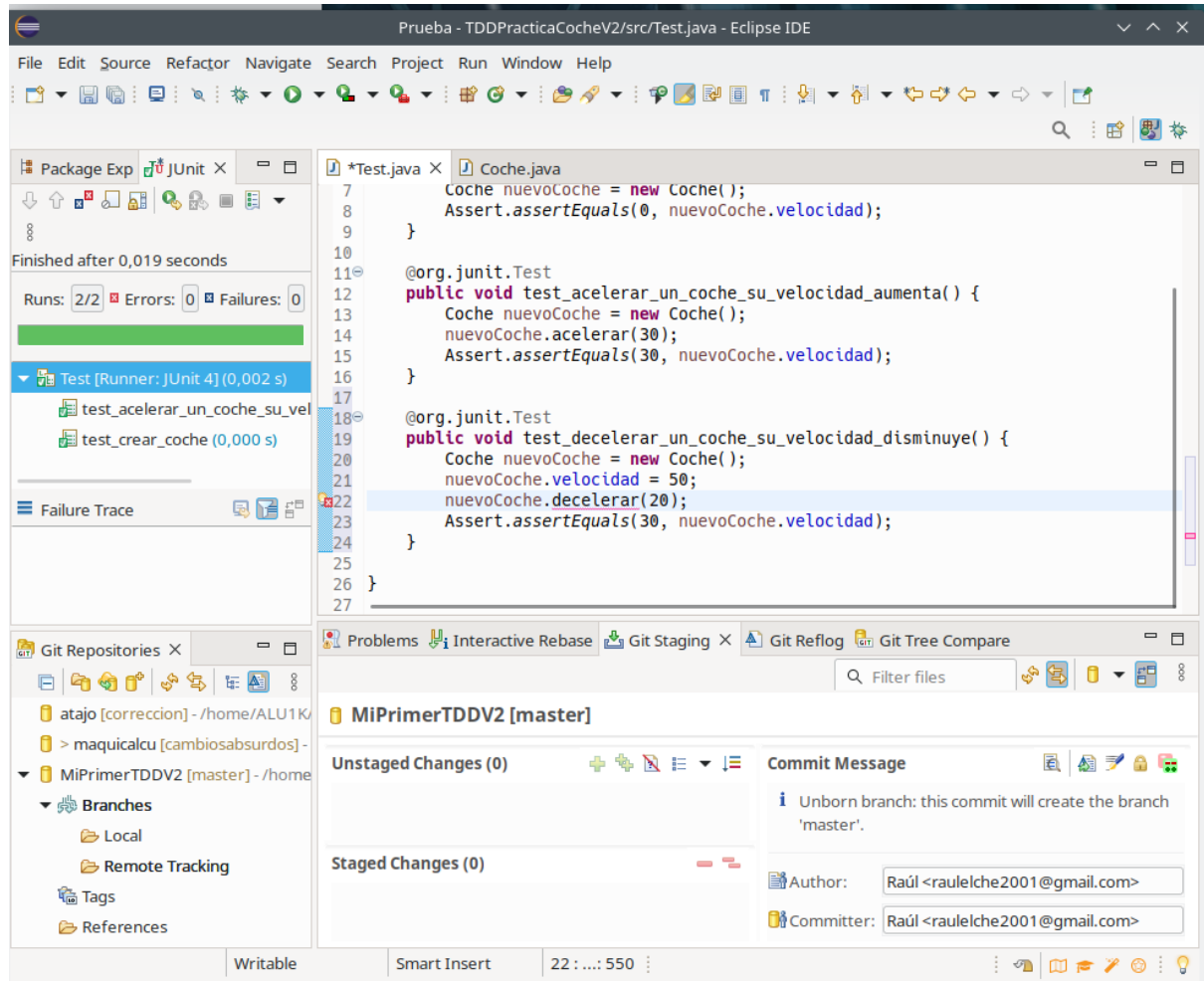
El parámetro de entrada se llamará “aceleracion” y este método sumará el valor que hemos pasado por parámetro al atributo velocidad y se le asignará a dicha variable.



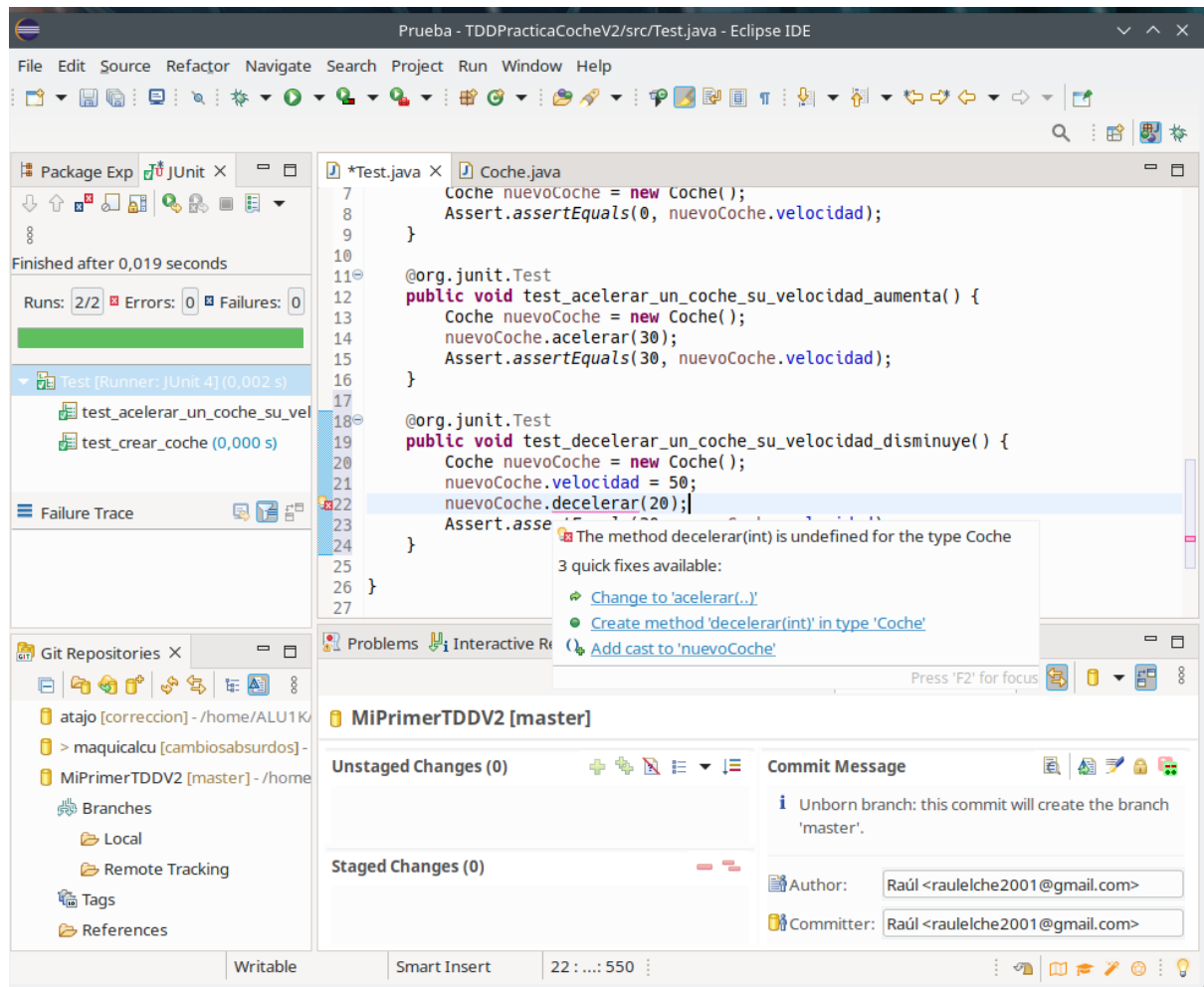
Ejecutamos el test y deberíamos pasarlo sin errores.



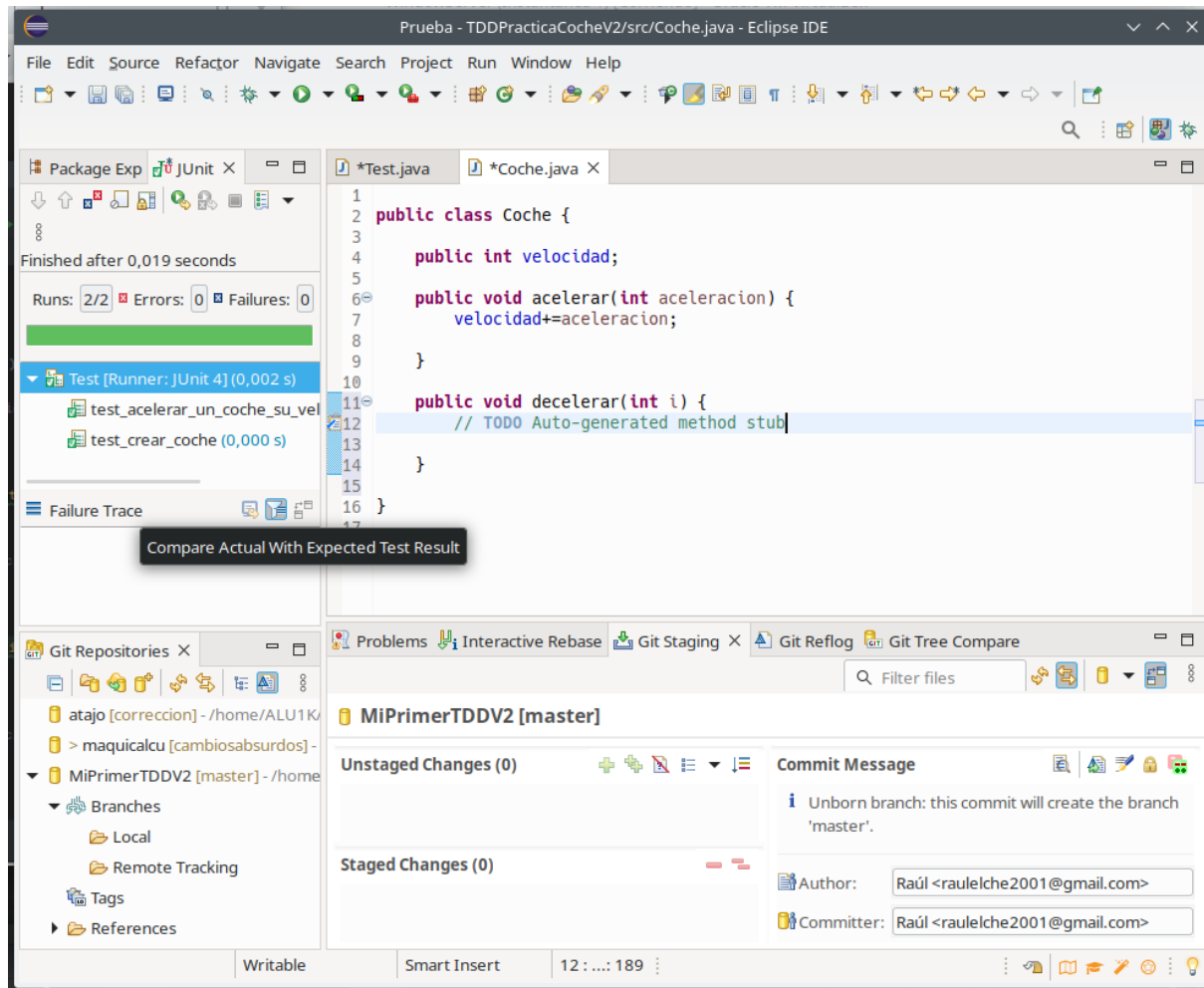
Creamos otro test, en este le asignaremos al coche una velocidad de 50 y esta vez el coche llamará a un método llamado decelerar que le pasaremos por parámetro un entero, en este caso 20, y tendrá que disminuir la velocidad del coche. Por lo tanto el método `Assert.equals` de este test tendrá que comprobar que la velocidad del coche al aplicar el método sea de 20.



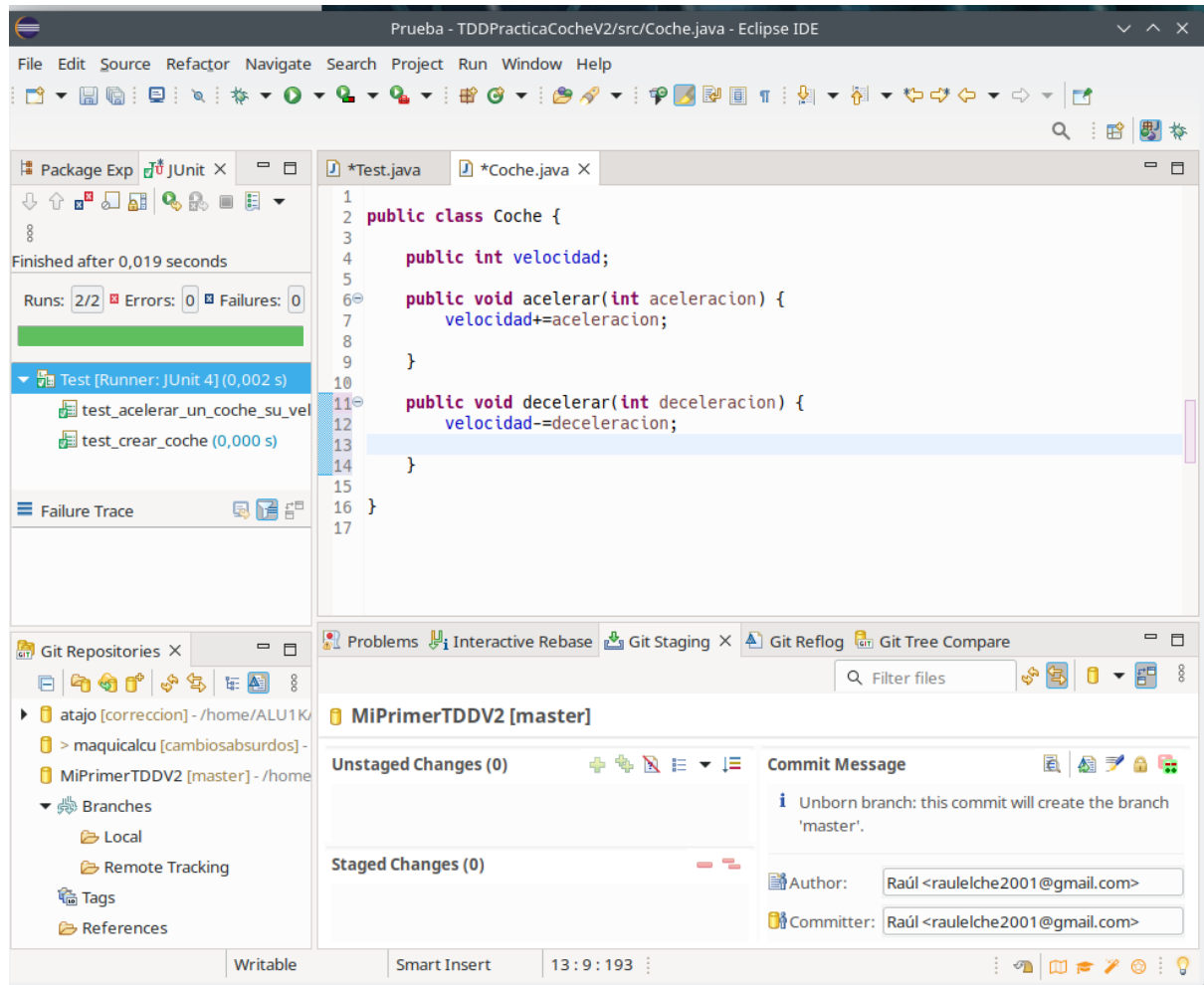
Este método decelerar aparece en rojo porque no existe, así que clicamos y nos colocamos encima de él y seleccionamos la opción “Create method ‘decelerar’ in type ‘Coche’”.



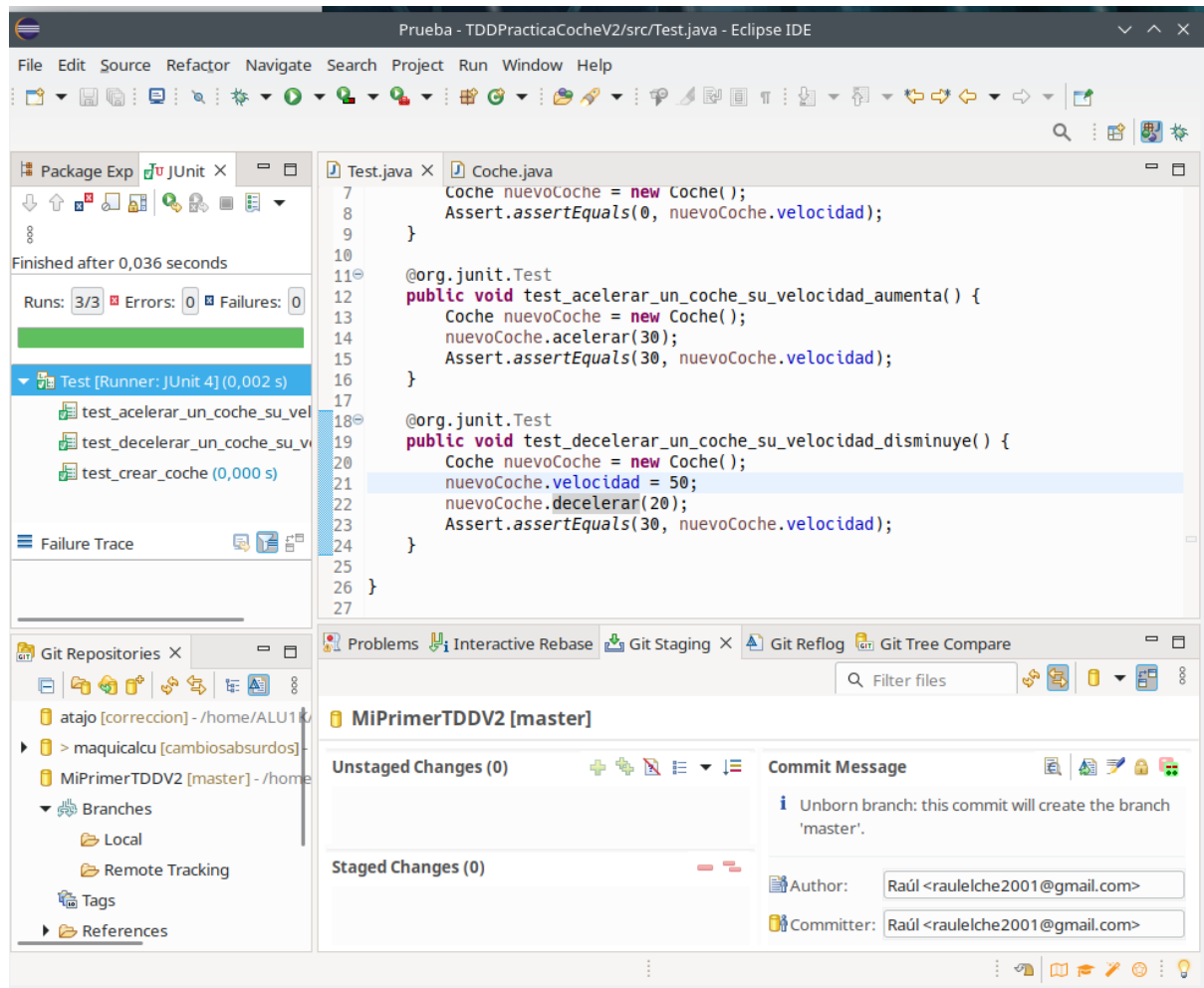
Una vez creado el método habrá que modificarlo.



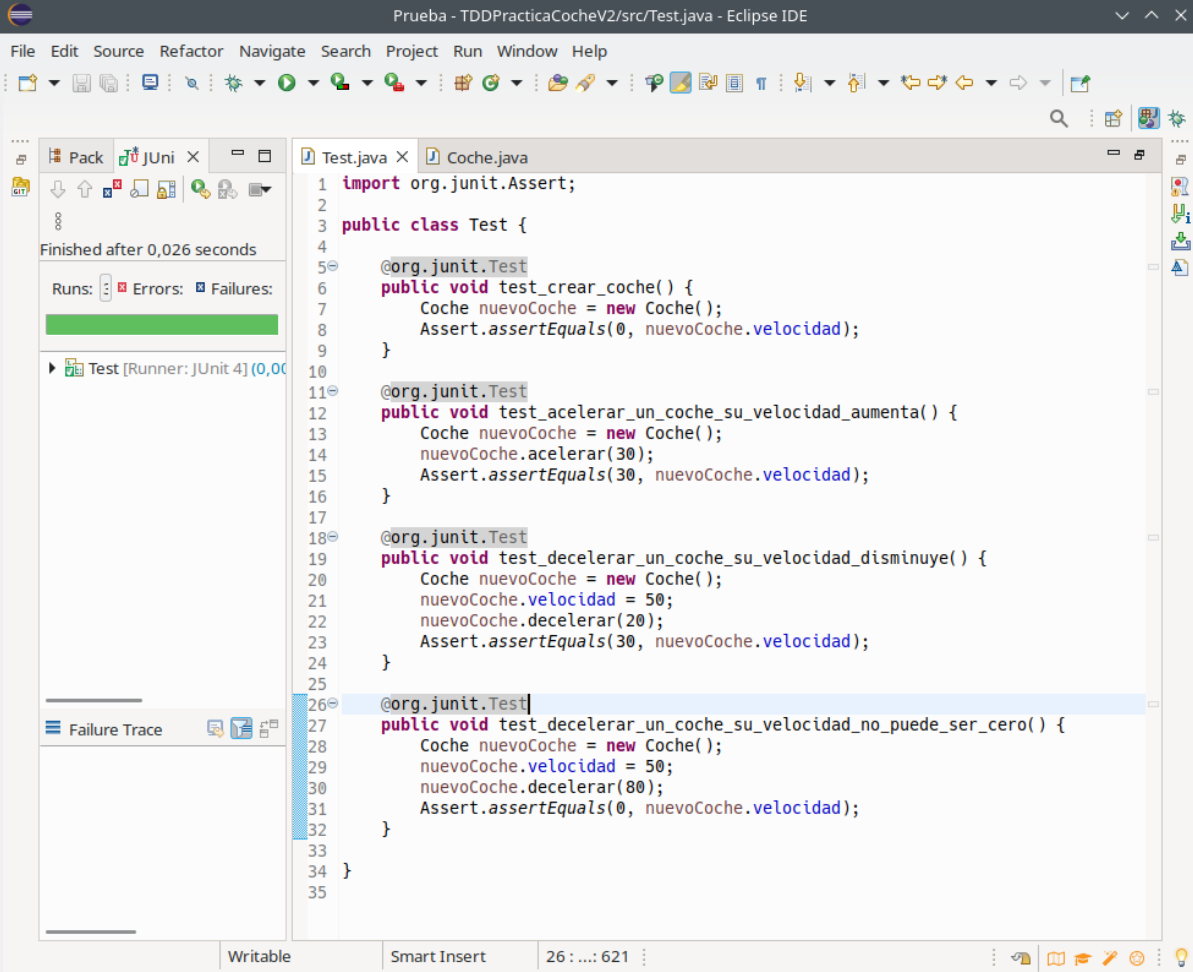
El parámetro de entrada se llamará “deceleracion” y este método restará el valor que hemos pasado por parámetro al atributo velocidad y se le asignará a dicha variable.



Ejecutamos el test y lo pasaremos sin errores.

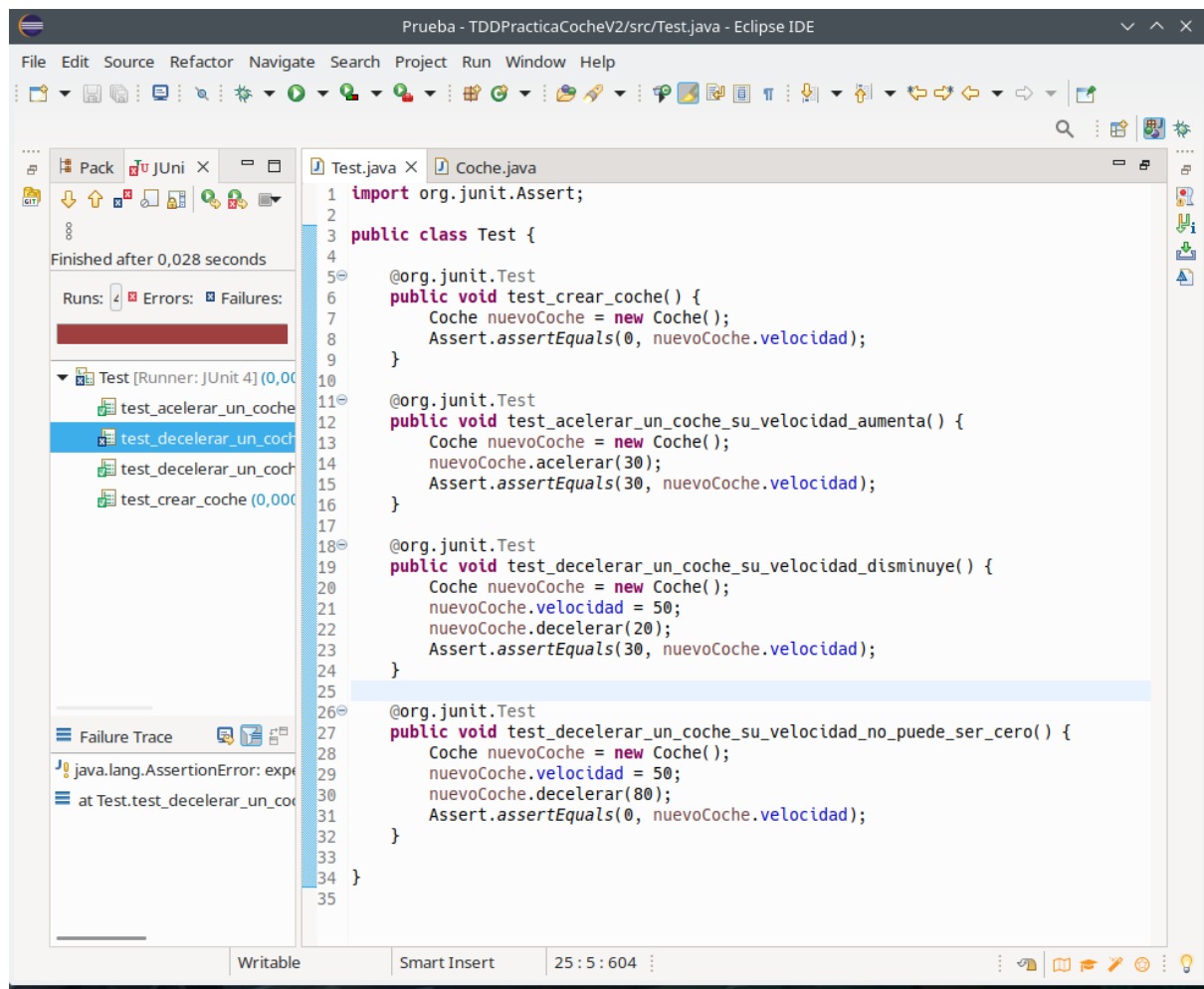


A continuación, creamos un último test en el que si la velocidad del coche disminuye más de lo que está acelerado deberá ser 0, por lo que, inicializamos la velocidad del coche a 50 y con el método decelerar la disminuimos 80. por lo que con el método Assert.equals debemos comprobar que la velocidad sea 0.

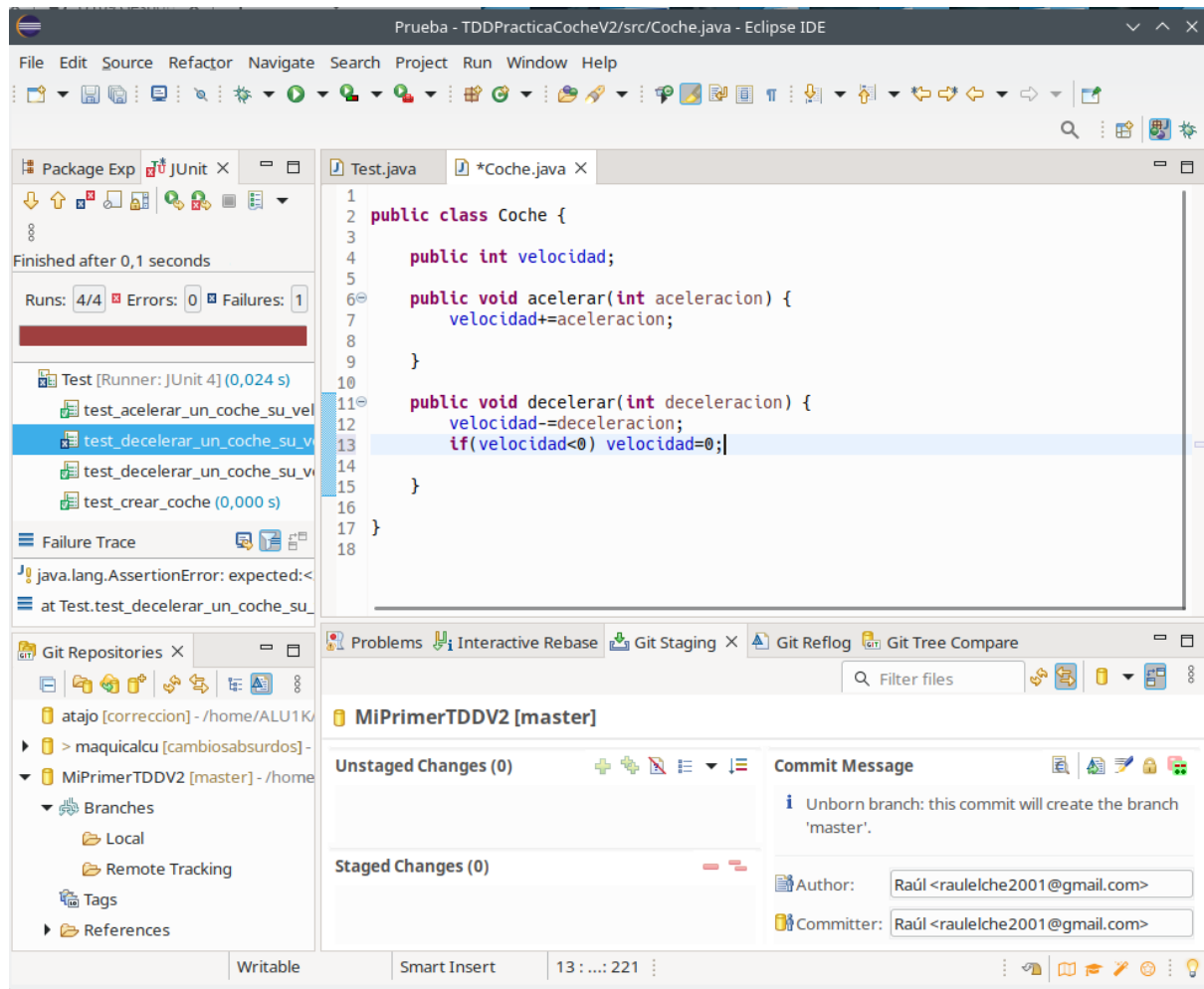


```
1 import org.junit.Assert;
2
3 public class Test {
4
5     @org.junit.Test
6     public void test_crear_coche() {
7         Coche nuevoCoche = new Coche();
8         Assert.assertEquals(0, nuevoCoche.velocidad);
9     }
10
11     @org.junit.Test
12     public void test_acelerar_un_coche_su_velocidad_aumenta() {
13         Coche nuevoCoche = new Coche();
14         nuevoCoche.acelerar(30);
15         Assert.assertEquals(30, nuevoCoche.velocidad);
16     }
17
18     @org.junit.Test
19     public void test_decelerar_un_coche_su_velocidad_disminuye() {
20         Coche nuevoCoche = new Coche();
21         nuevoCoche.velocidad = 50;
22         nuevoCoche.decelerar(20);
23         Assert.assertEquals(30, nuevoCoche.velocidad);
24     }
25
26     @org.junit.Test
27     public void test_decelerar_un_coche_su_velocidad_no_puede_ser_cero() {
28         Coche nuevoCoche = new Coche();
29         nuevoCoche.velocidad = 50;
30         nuevoCoche.decelerar(80);
31         Assert.assertEquals(0, nuevoCoche.velocidad);
32     }
33 }
34
35
```

Si ejecutamos el test fallará, porque en el método decelerar no tenemos ninguna estructura que controle esto.



En el método creamos un condicional, en el caso de que la velocidad sea menor que 0, se le asignará el valor 0 la velocidad.



Si ahora ejecutamos el test lo pasaremos sin errores.

