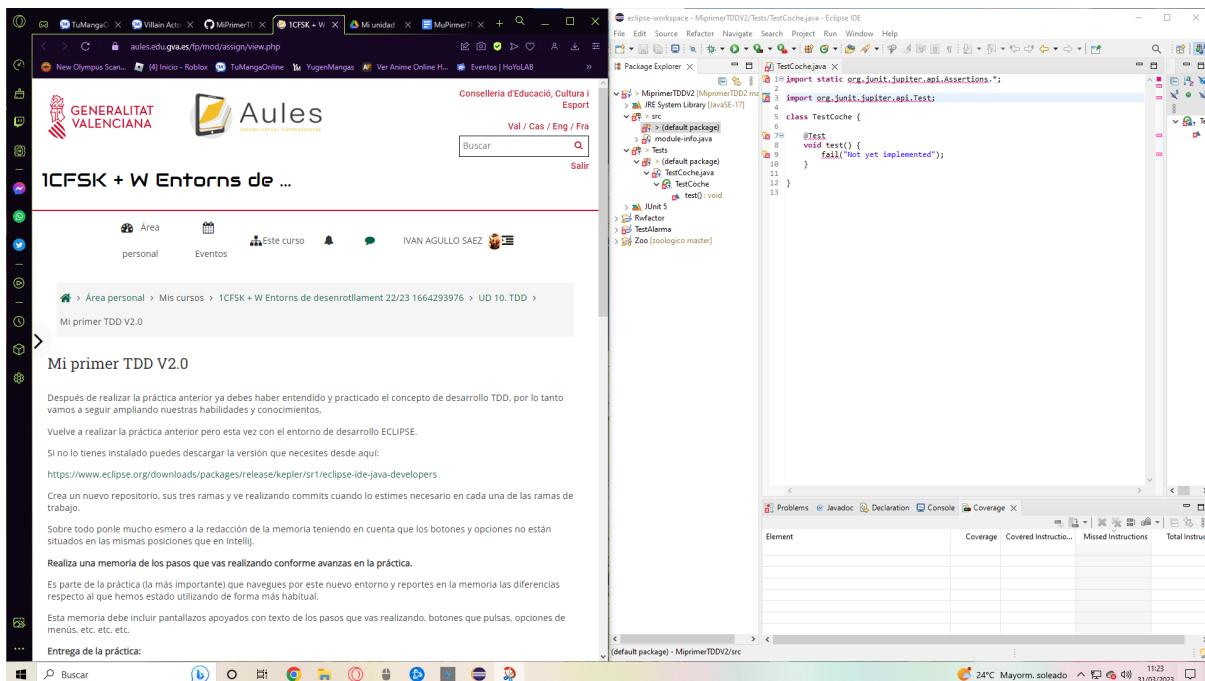
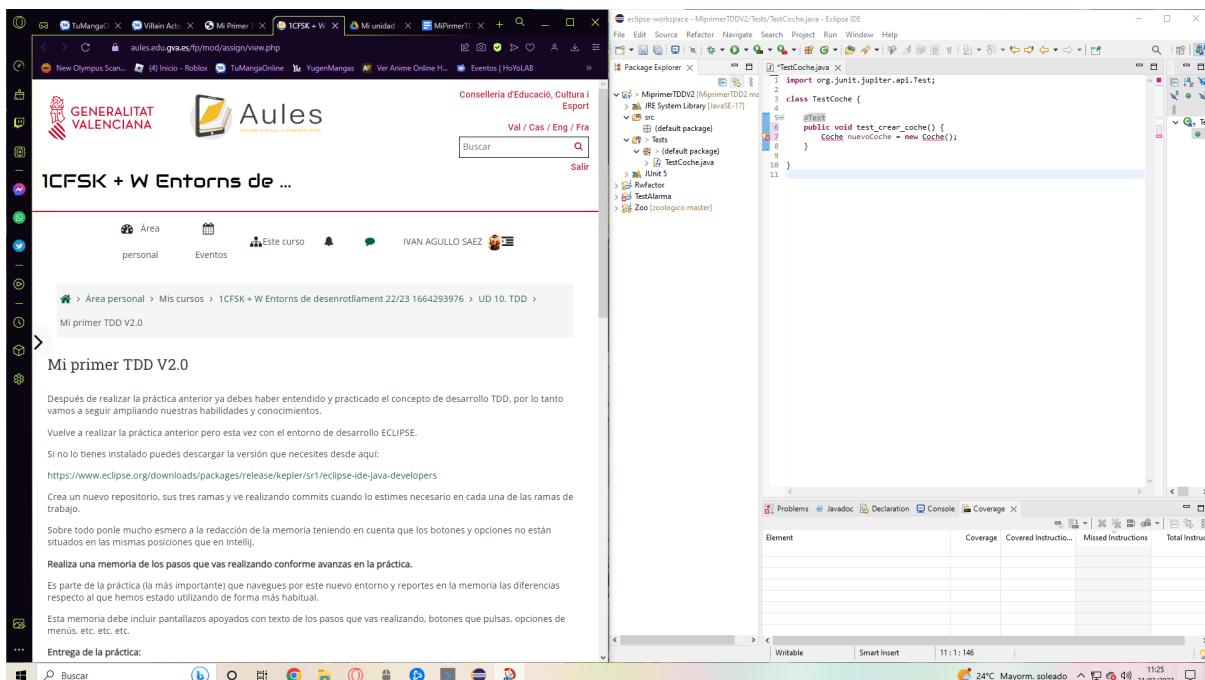


# Mi Primer TDD v2

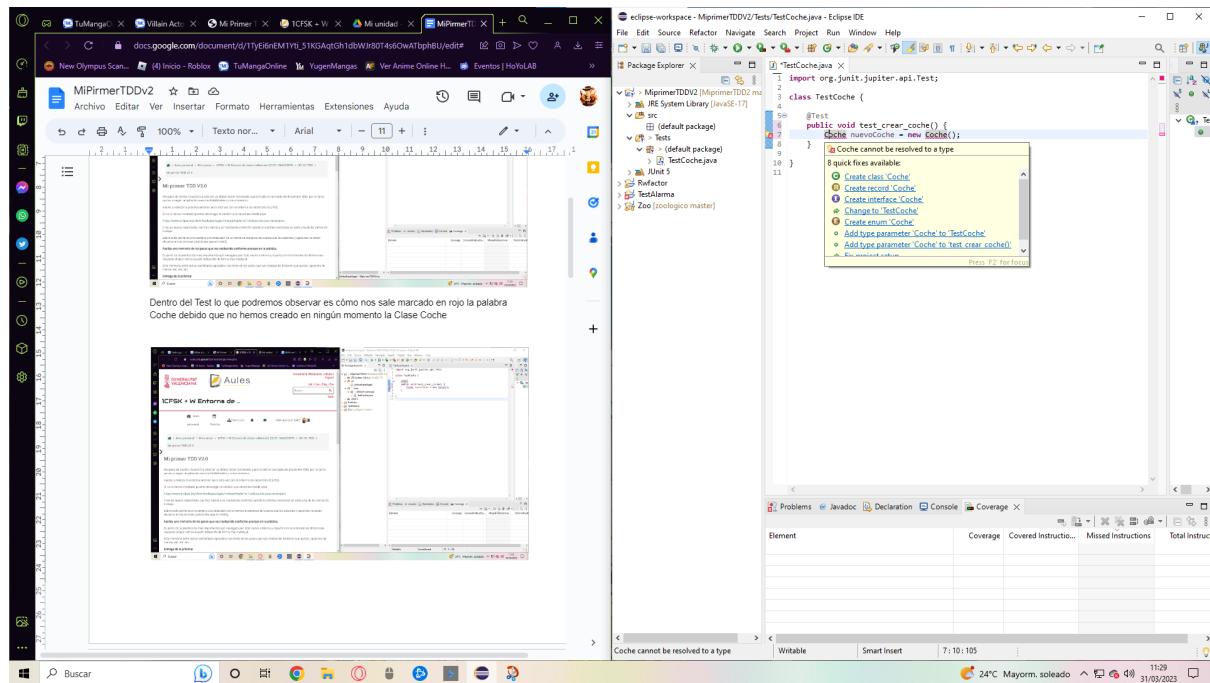
Creamos un nuevo proyecto. Creamos una carpeta llamada Tests y luego hacemos click derecho > New > Junit Test Case. Le pondremos de nombre TestCoche.



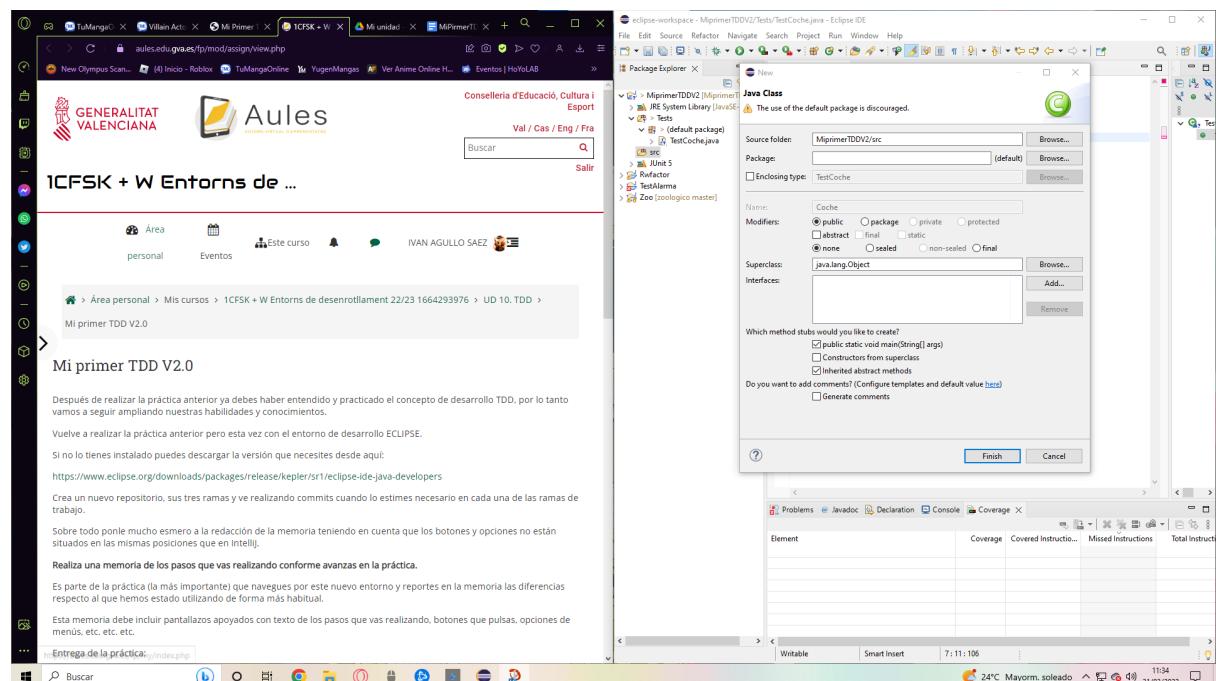
Dentro del Test lo que podremos observar es cómo nos sale marcado en rojo la palabra Coche debido que no hemos creado en ningún momento la Clase Coche



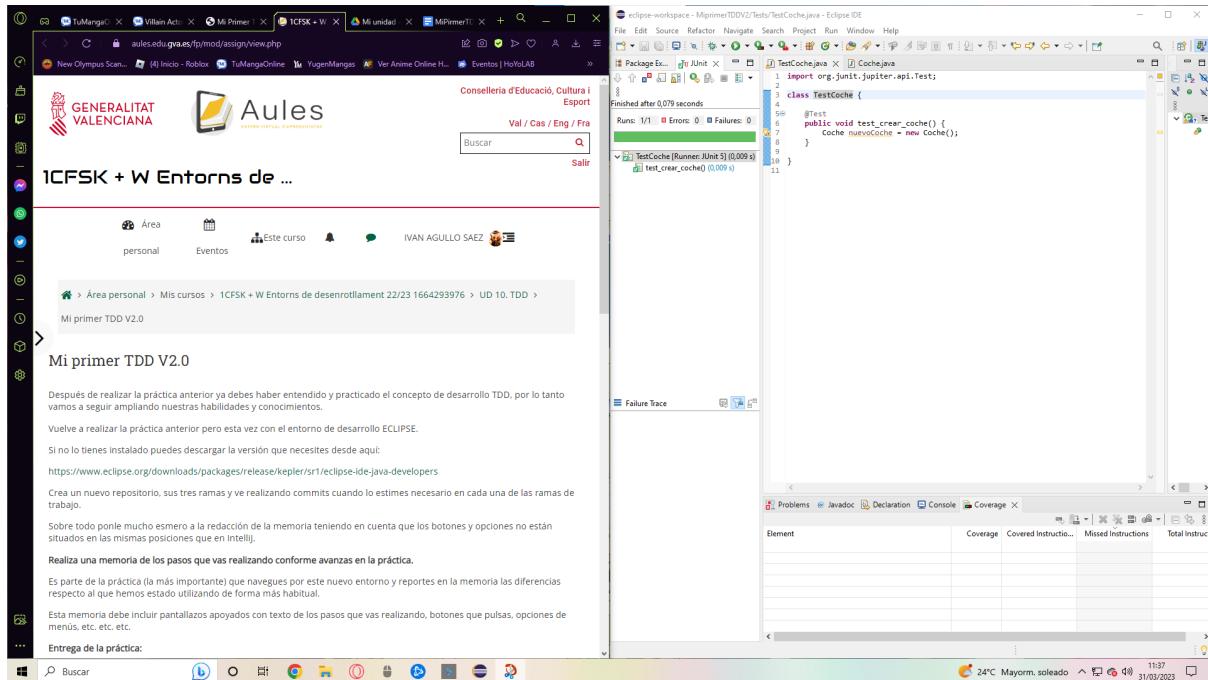
Si hacemos clic en Coche nos aparecerá una pestaña con la cual podremos crear rápidamente la clase Coche.



Cuando vayamos a crear la clase Coche debemos de indicarle que la queremos en la carpeta src.

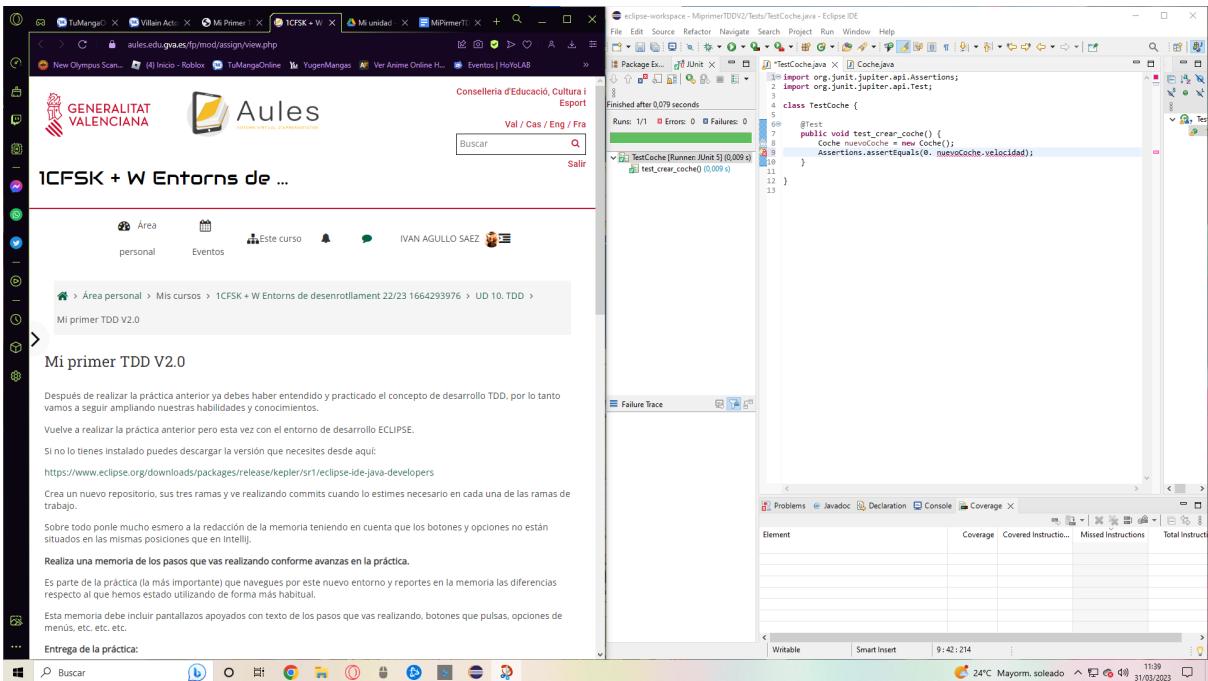


Ahora si ejecutamos la clase TestCoche veremos como funciona.

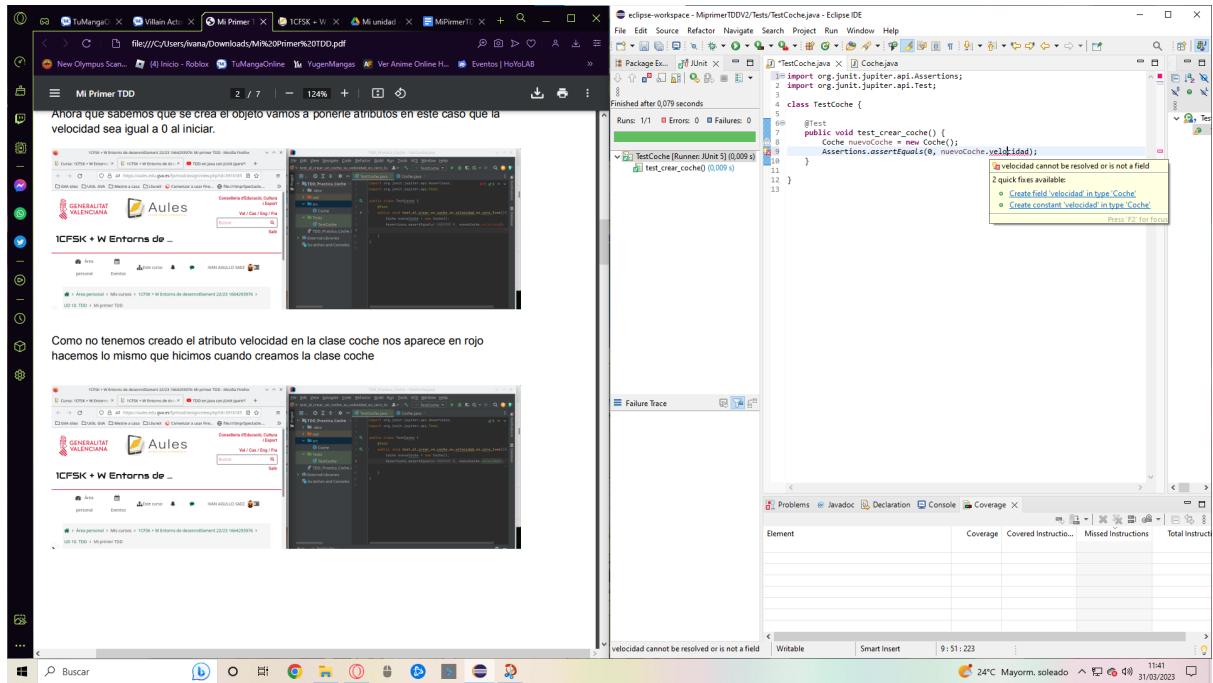


Ahora que sabemos que se crea el objeto vamos a ponerle atributos en este caso que la velocidad sea igual a 0 al iniciar.

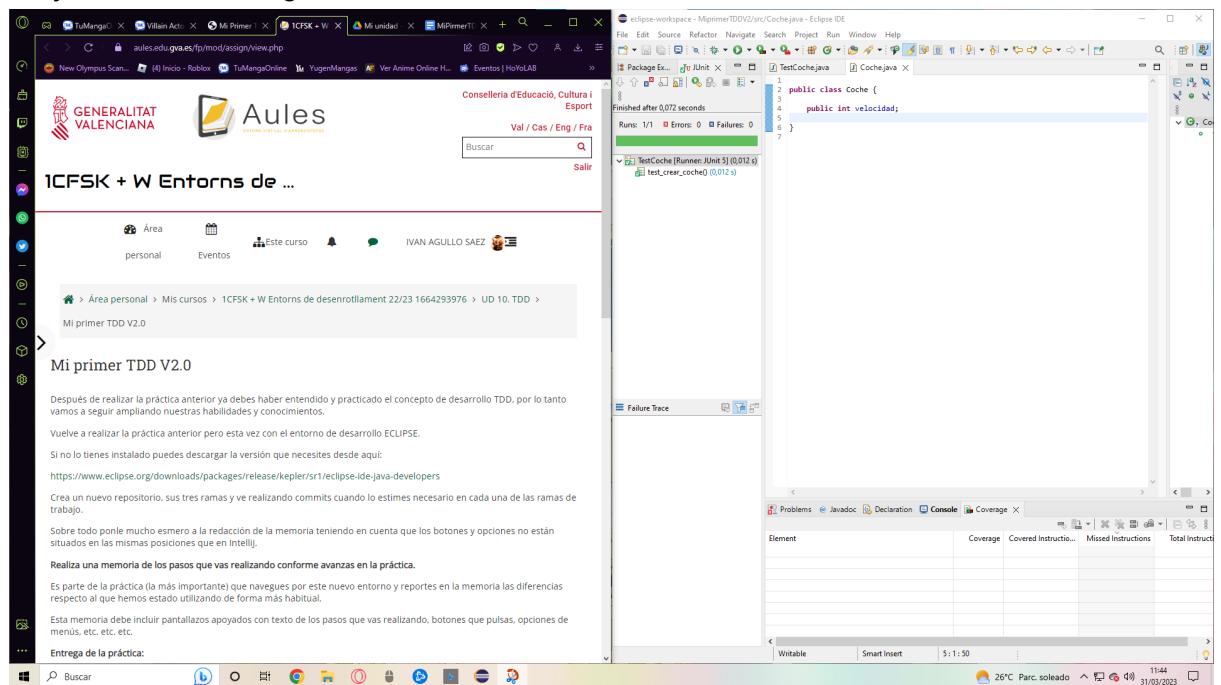
Como no tenemos creado el atributo velocidad nos aparecerá en rojo.



Igual que hicimos con la clase coche, podemos hacerlo con el atributo velocidad.

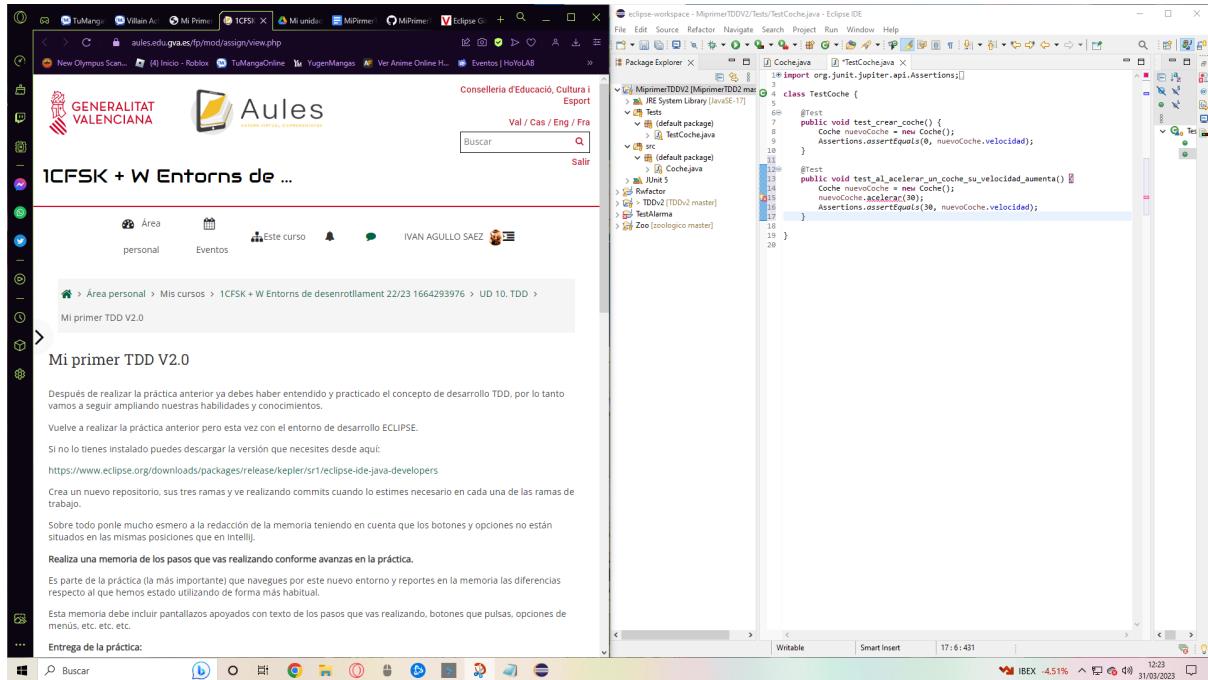


Si ejecutamos el código veremos como funciona.

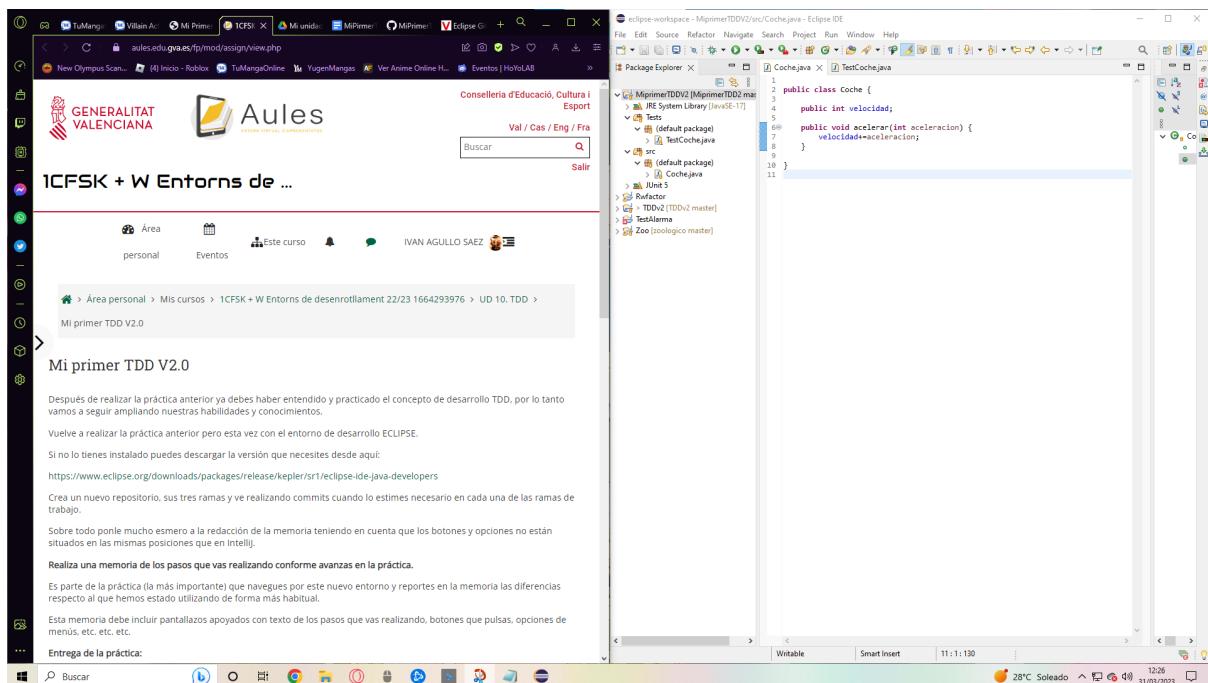


Ahora vamos a añadir más test. Vamos a hacer uno para que el coche pueda acelerar, para ello puedes copiar el anterior y modificarlo. Añade una línea con una llamada al método acelerar() que esperara un int. Y luego que compruebe que la velocidad es igual.

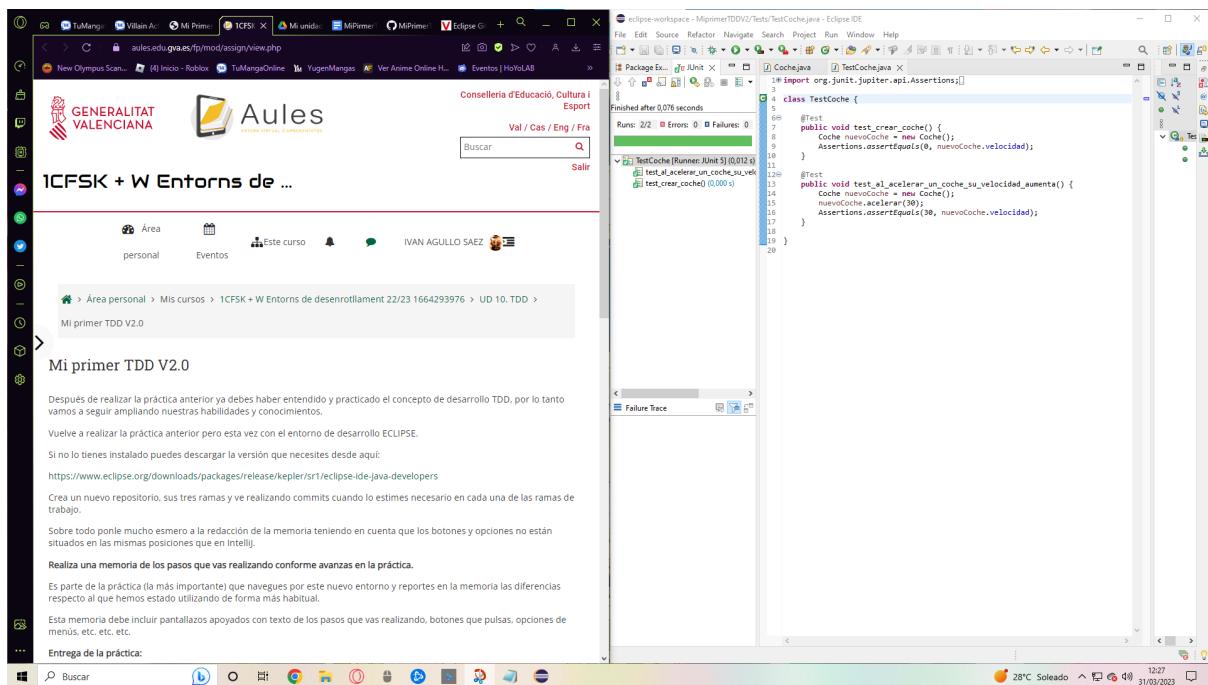
También tendremos que añadir el método acelerar a la clase Coche



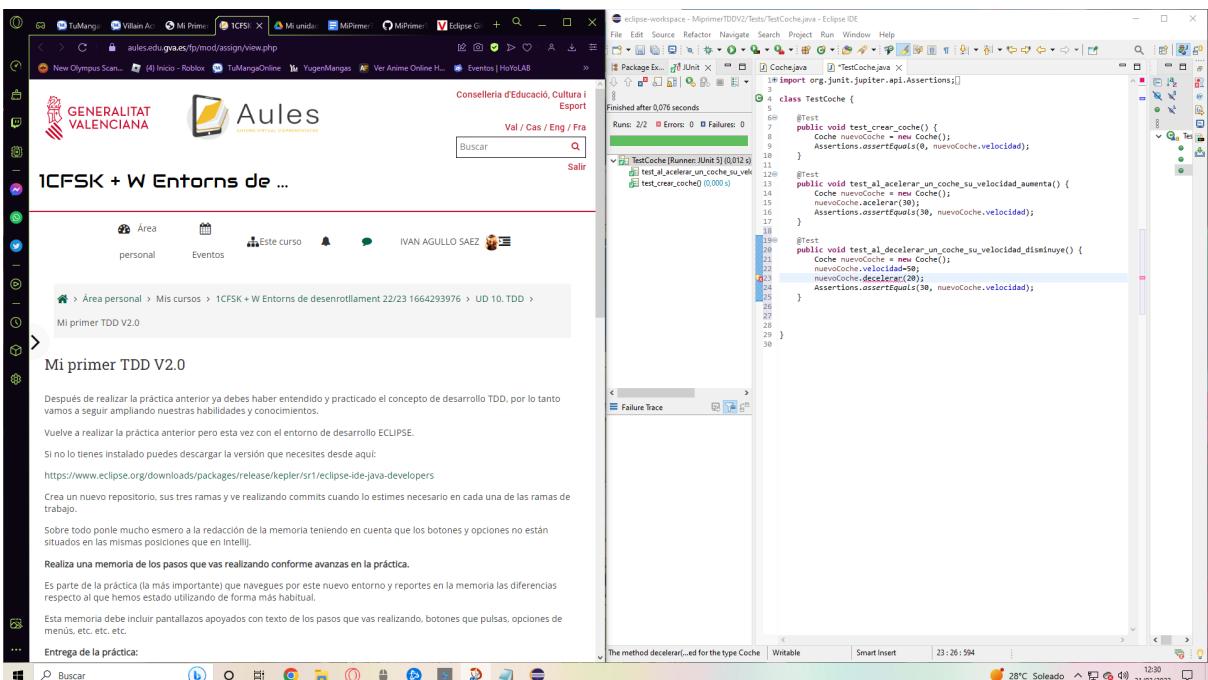
Ahora si nos vamos a la clase Coche tendremos el método acelerar, el cual tendremos que hacer que velocidad sea igual a velocidad más aceleración.



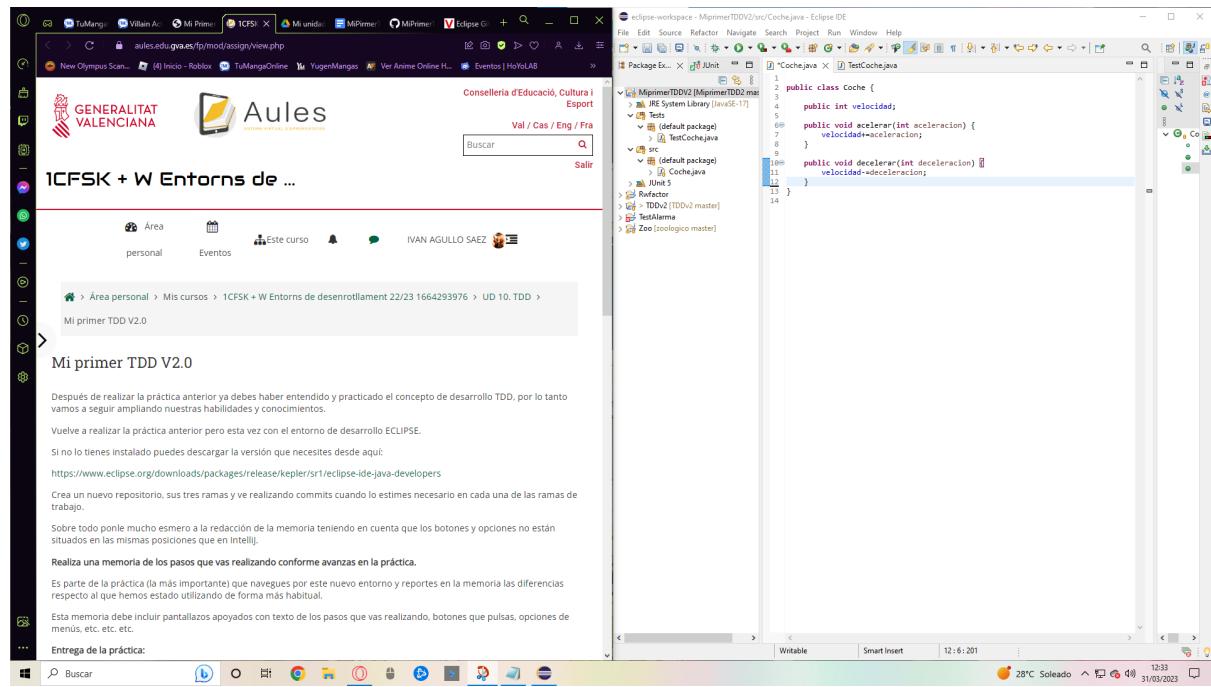
Si ejecutamos ahora el código veremos cómo todo funciona.



Ahora vamos a hacer otro test en este caso para decelerar, para ello copiamos el anterior y modificamos el nombre del test y cambiamos el llamado al método por decelerar que será un nuevo método. Lo mismo que antes nos vamos a la clase y creamos el método decelerar.

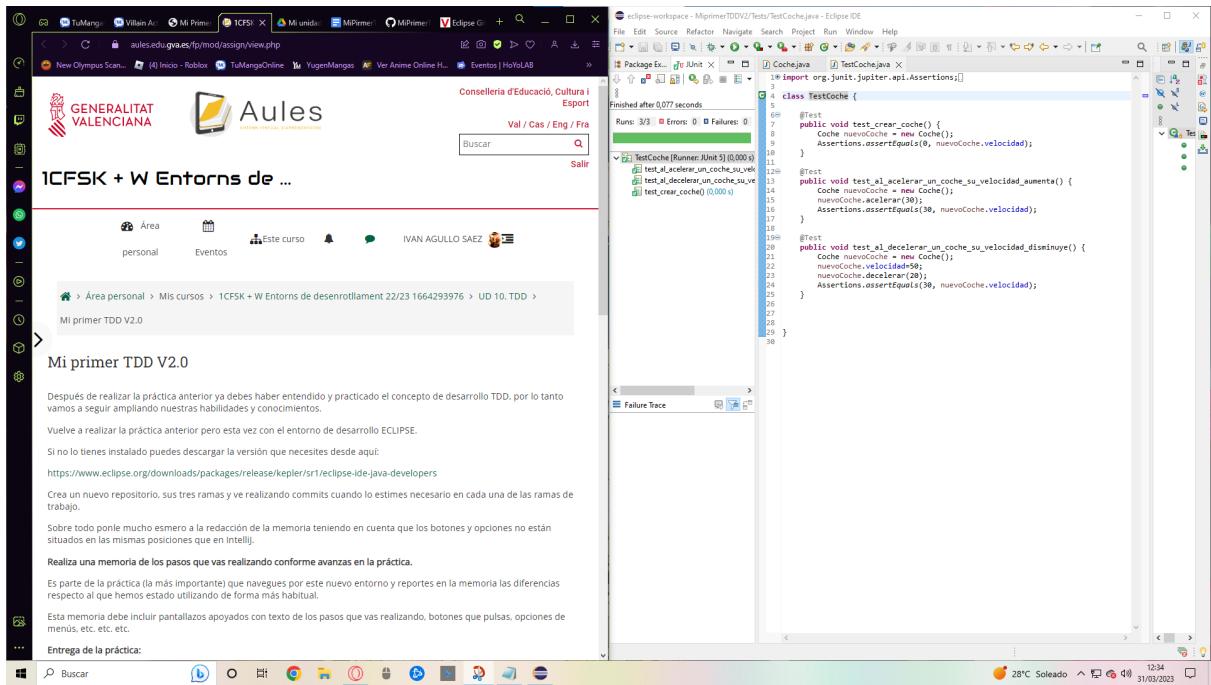


Ahora en la clase Coche tendremos el método decelerar que será lo mismo que acelerar pero restando.



```
1 package MiPrimerTDDV2;
2
3 public class Coche {
4     public int velocidad;
5
6     public void acelerar(int aceleracion) {
7         velocidad+=aceleracion;
8     }
9
10    public void decelerar(int deceleracion) {
11        velocidad-=deceleracion;
12    }
13
14 }
```

Ahora si ejecutamos TestCoche veremos como funcionan todos los test.



```
1 package MiPrimerTDDV2.Tests;
2
3 import org.junit.jupiter.api.Assertions;
4
5 class TestCoche {
6
7     @Test
8     public void test_crear_coche() {
9         Coche nuevoCoche = new Coche();
10        Assertions.assertEquals(0, nuevoCoche.velocidad);
11    }
12
13    @Test
14    public void test_acelerar_un_coche_su_velocidad_aumenta() {
15        Coche nuevoCoche = new Coche();
16        nuevoCoche.acelerar(30);
17        Assertions.assertEquals(30, nuevoCoche.velocidad);
18    }
19
20    @Test
21    public void test_el_decelerar_un_coche_su_velocidad_disminuye() {
22        Coche nuevoCoche = new Coche();
23        nuevoCoche.decelerar(30);
24        Assertions.assertEquals(30, nuevoCoche.velocidad);
25    }
26
27
28
29 }
```

Pero si la velocidad de decelerar es mayor que la velocidad que hay en ese momento, nos dará un error.

The screenshot shows a dual-monitor setup. On the left monitor, a web browser displays a Moodle course page for '1CFSK + W Entorns de ...'. The page includes navigation links like 'Área personal', 'Eventos', and 'IVAN AGULLO SAEZ'. Below the navigation is a text area with instructions and links. On the right monitor, an Eclipse IDE window is open, showing the code for a 'Coche' class and its corresponding test cases ('TestCoche'). The code includes methods for accelerating and decelerating a car, with assertions for each. The Eclipse interface shows various toolbars and panels, and the status bar at the bottom indicates the date and time.

Para solucionar el error debemos de irnos al método decelerar de la clase coche. con un if haremos que si la velocidad es menor a 0, la velocidad será igual a 0.

Con esto si la velocidad de decelerar es mayor a la velocidad actual será 0 es decir se habrá frenado.

This screenshot shows the same dual-monitor setup as the previous one. The left monitor displays the Moodle course page, and the right monitor shows the Eclipse IDE with the modified Java code for the 'Coche' class. The changes made to the code are:

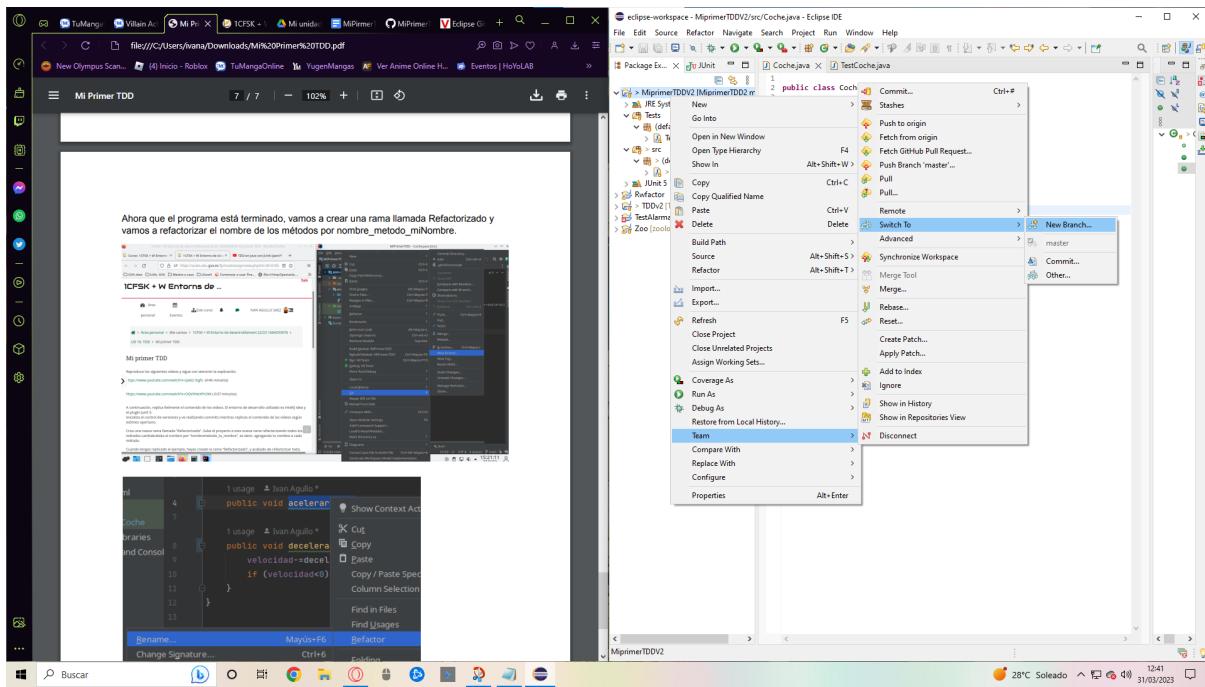
```

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;
    }
}

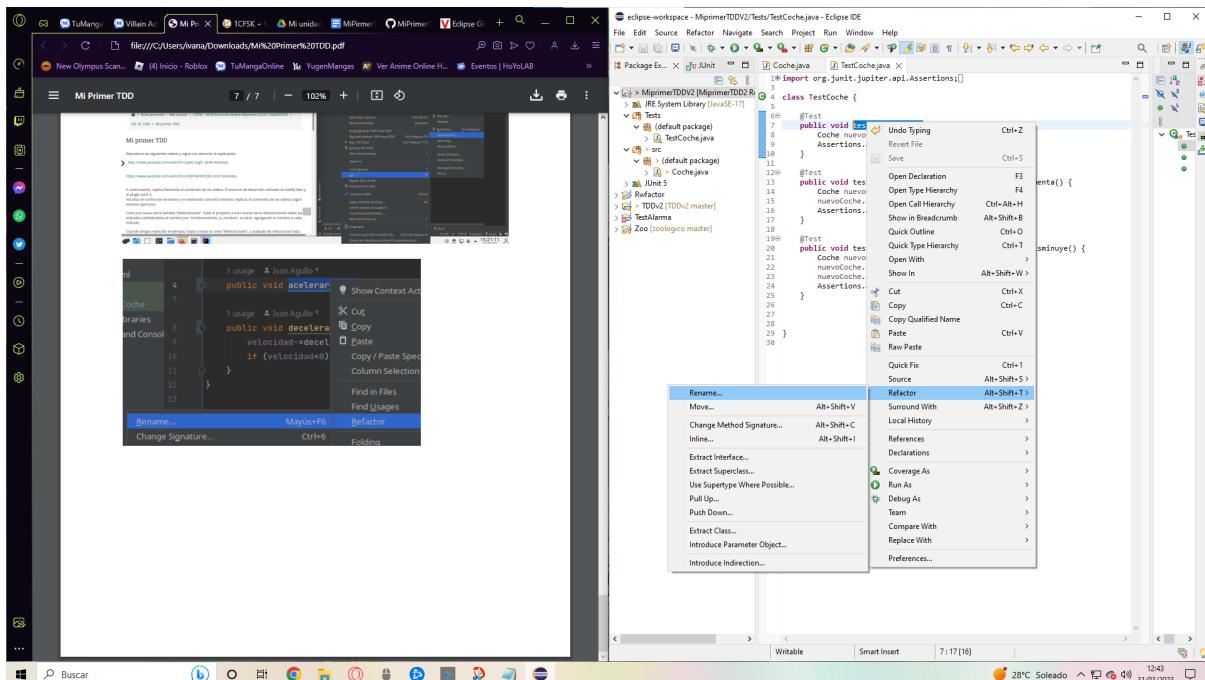
```

The Eclipse interface remains similar to the first screenshot, with various toolbars and panels visible.

Ahora vamos a cerrar una nueva rama llamada Refactorizado.



Nos situamos en la rama recién creada y nos ponemos a refactorizar todos los métodos para que queden terminen con tu nombre.

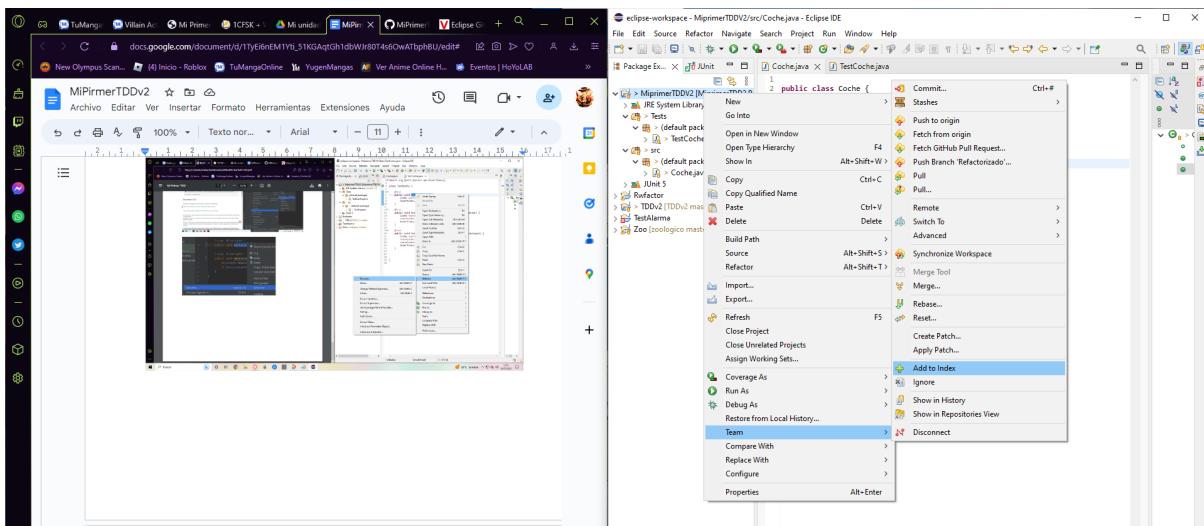


# Hacer un Commit y un push.

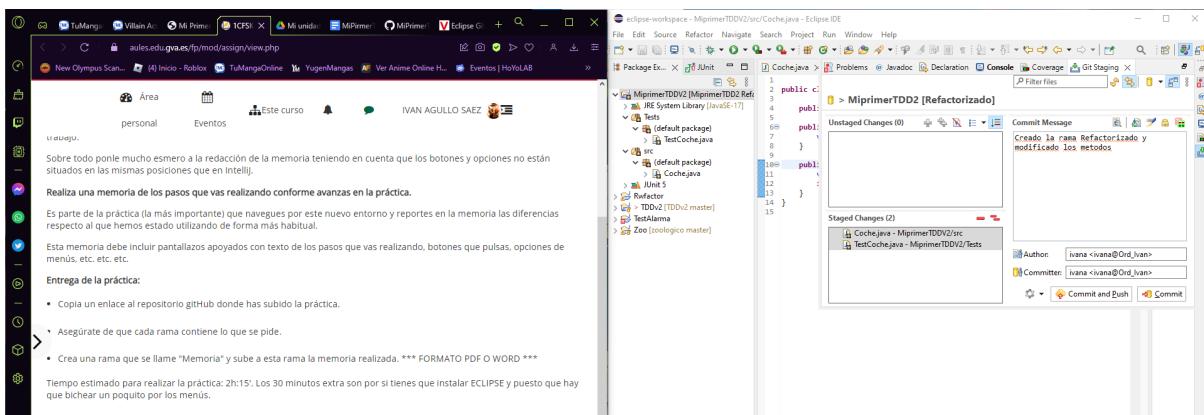
Para poder hacer un commit primero deberás compartir el repositorio con tu repositorio remoto de github mediante el code.

Cuando tengas creado ya algo en el repositorio y quieras hacer un commit, primero deberás hacer un Add to Index porque si no, no te dejará.

(NT: Estuve sin saber eso como 20 minutos pero al final lo saqué.)



Una vez hecho el anterior paso, te vas a Commit y puedes realizar el commit. Puedes realizar todos los commits primero y después enviarlo o hacer un commit y enviarlo al momento.



A la hora de hacer un push, ten encuesta que deberás marcar que cuando lo hagas sea un rebase y no un merge ya que queremos las ramas separadas.

También cuando haces un push solo envías la rama seleccionada(Por defecto es en la que te encuentres en ese momento).

