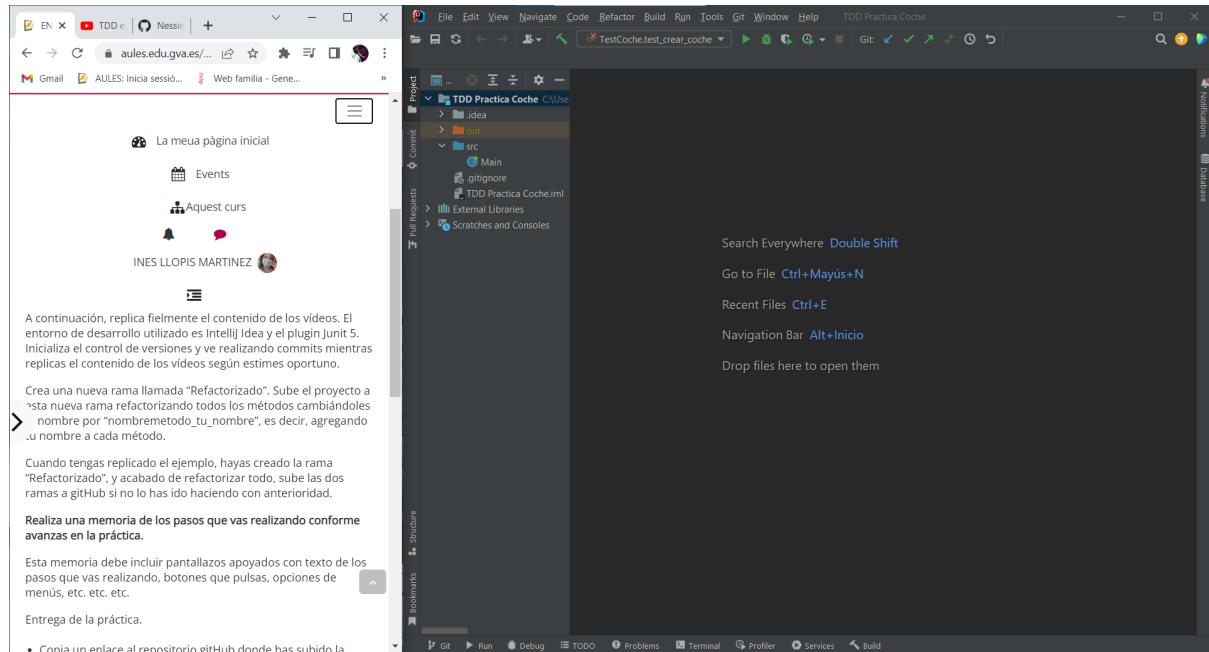


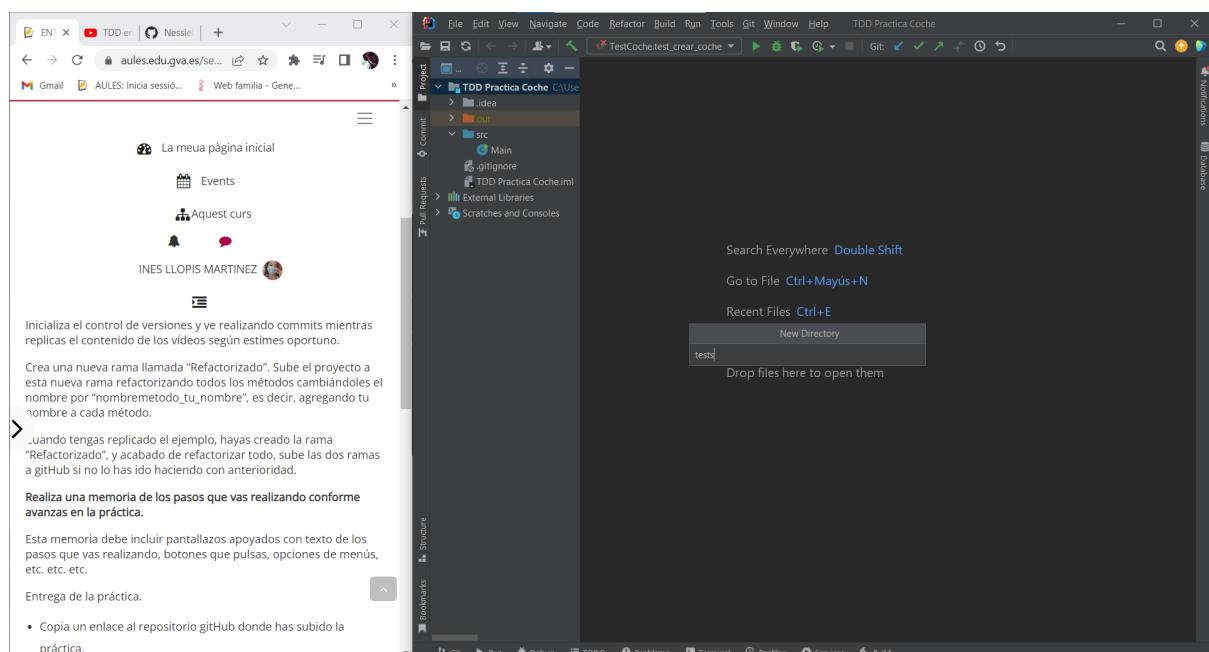
Entornos de Desarrollo

Ejercicio para entregar UD10

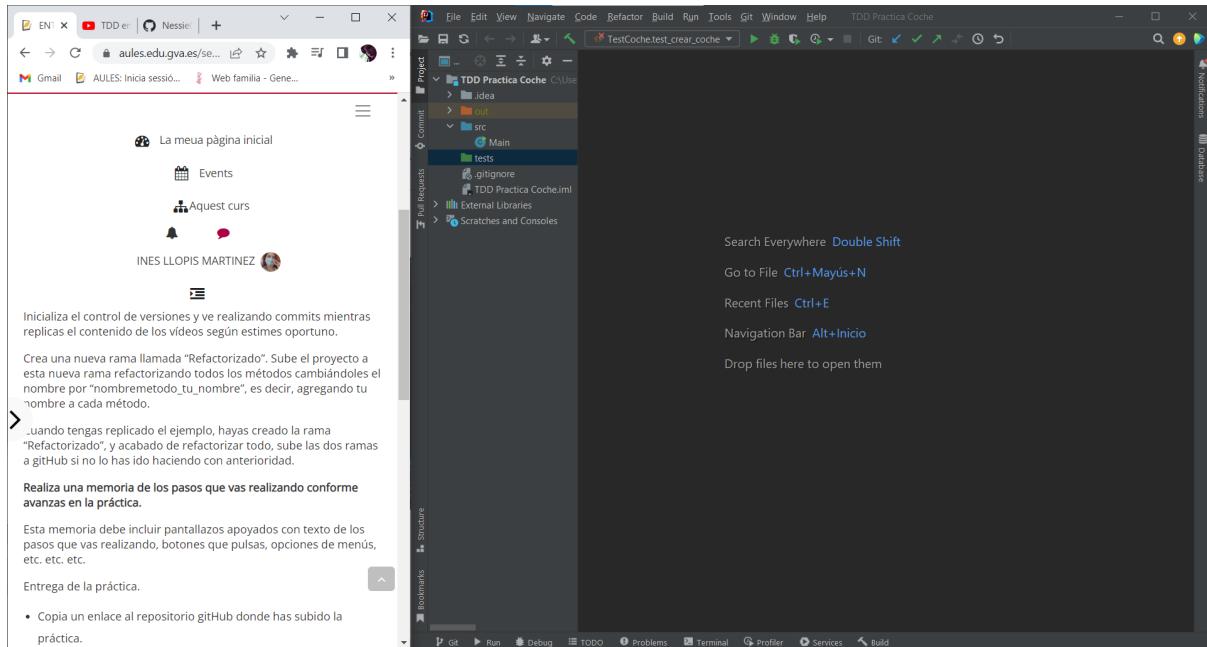
Iniciamos la práctica creando un proyecto llamado “TDD Practica Coche” en el entorno IntelliJ. Para ello hacemos clic en “File > New Project...”, le damos el nombre que nos interesa y hacemos clic en “Ok”. Realizamos un primer commit.



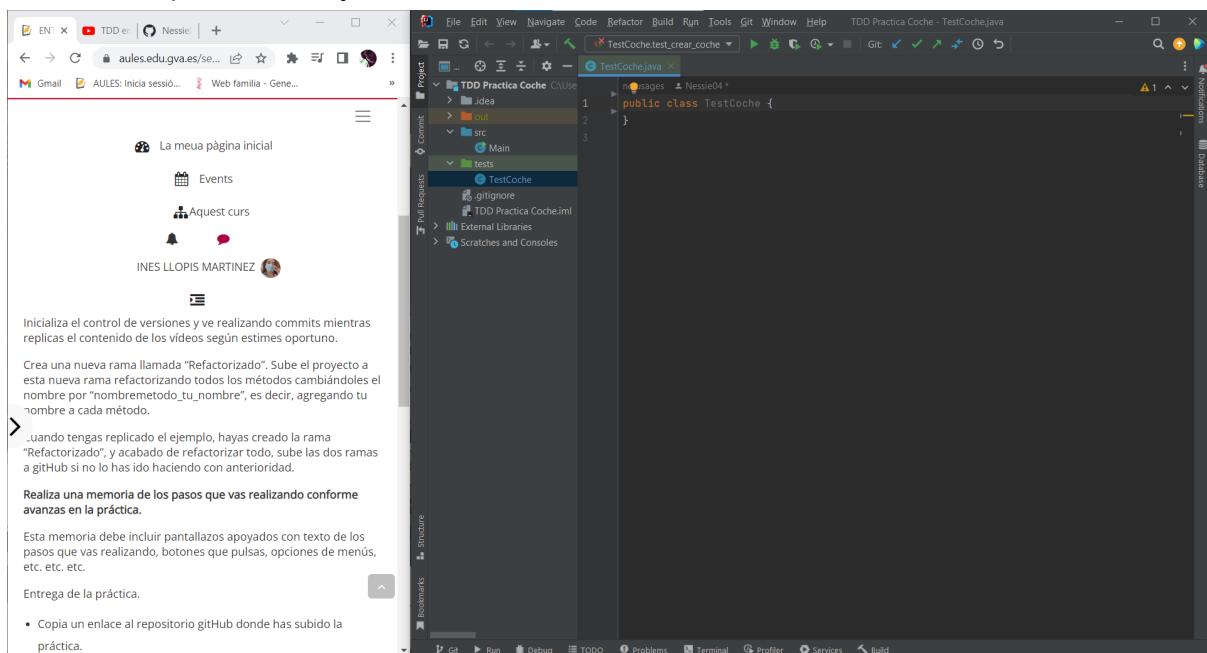
A continuación, creamos un directorio llamado “tests” dentro del proyecto. Para ello hacemos clic derecho sobre el proyecto y seleccionamos “New... > Directory”. Hacemos commit.



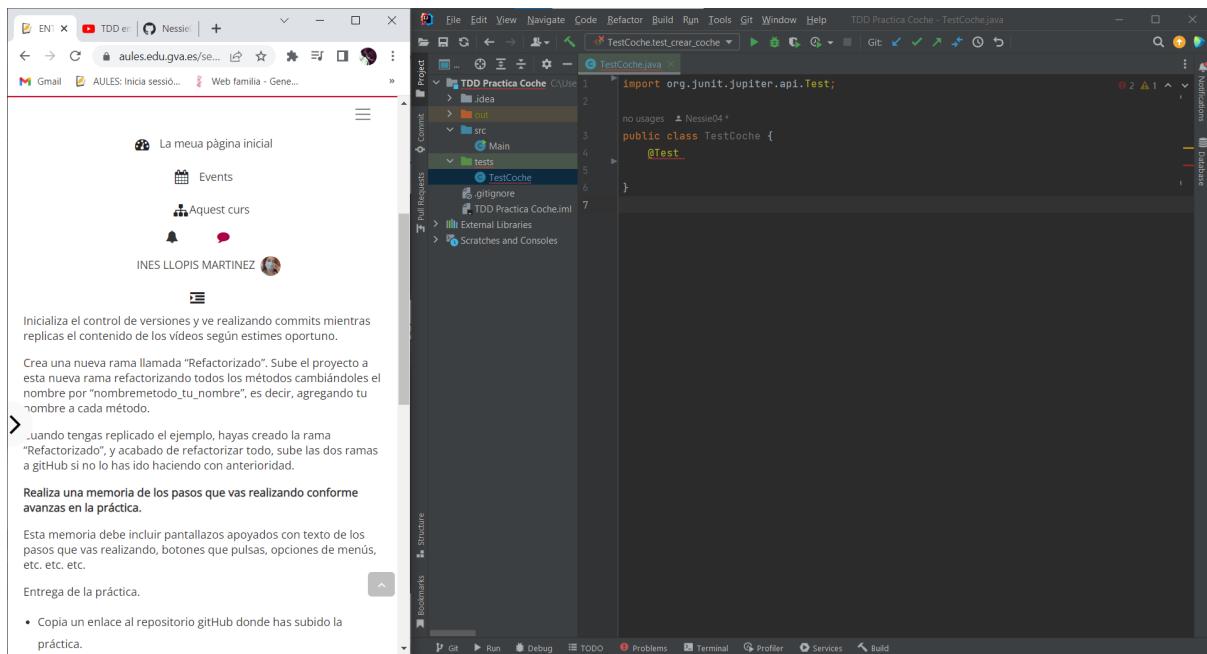
Hacemos clic derecho sobre el directorio que acabamos de crear, bajamos al final de la lista de opciones, seleccionamos “Mark directory as > Test Sources Root”. La carpeta del directorio cambiará de color azul a verde.



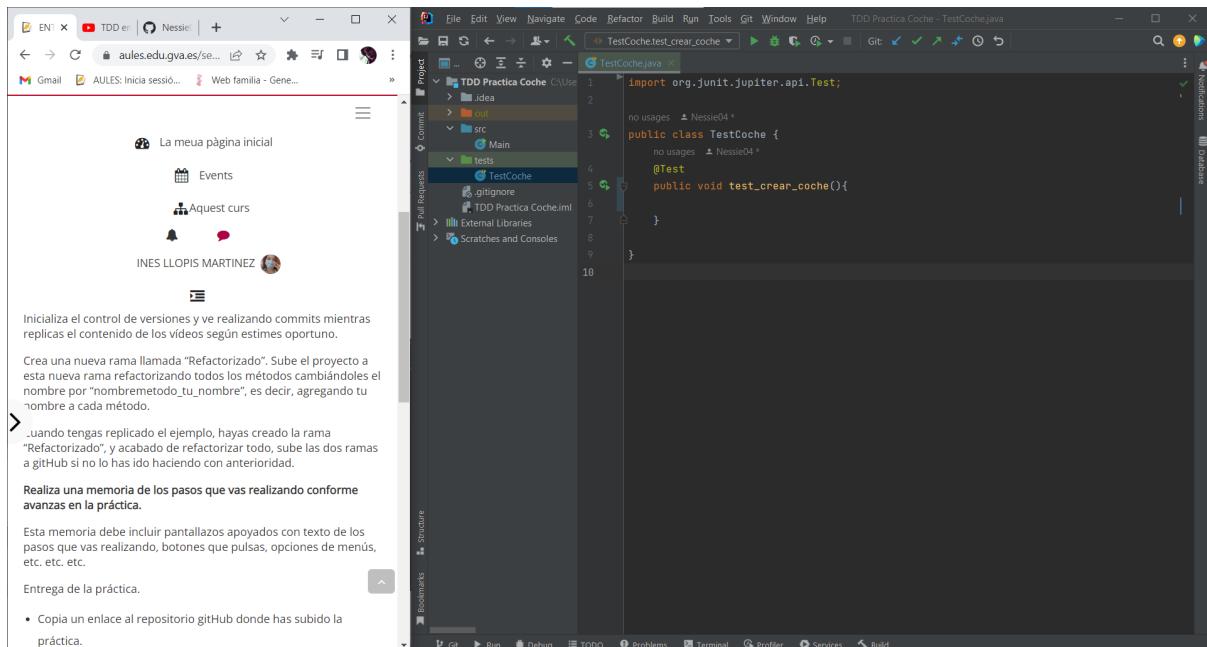
Creamos una clase dentro del directorio llamada “TestCoche” haciendo clic derecho sobre la carpeta “tests” y seleccionando “New... > JavaClass”. Hacemos commit.



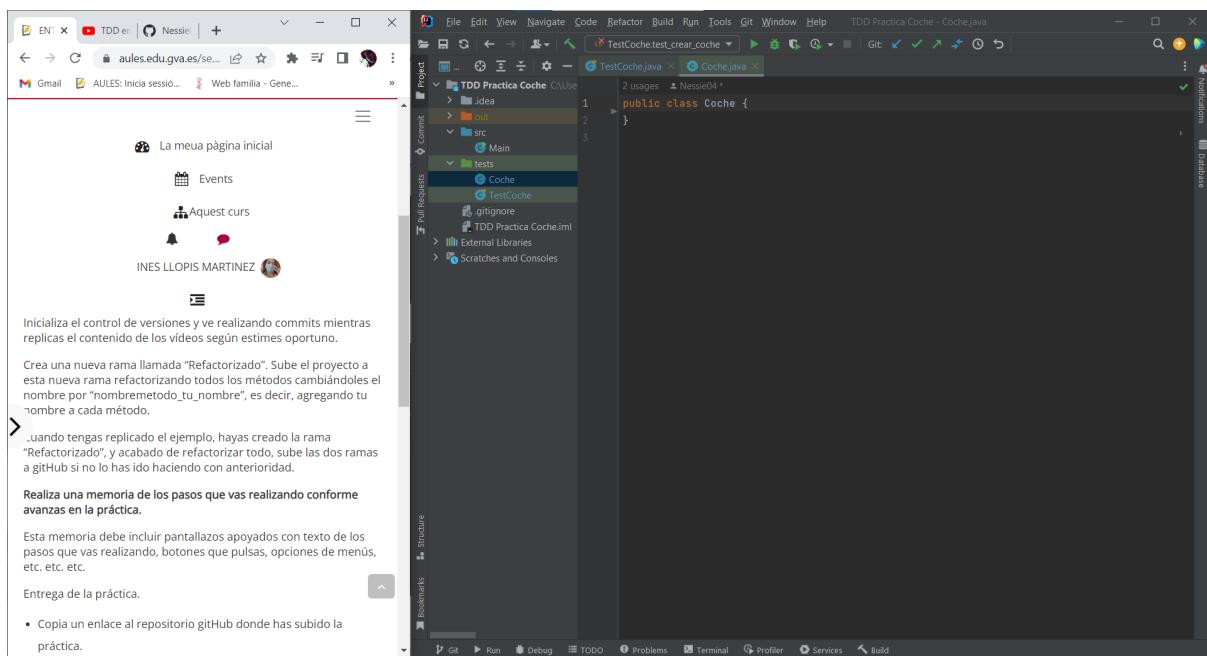
Escribimos @Test dentro de la clase TestCoche. Mantenemos el cursor encima de la palabra subrayada en rojo hasta que aparezca la opción de importar JUnit y seleccionamos JUnit 5.



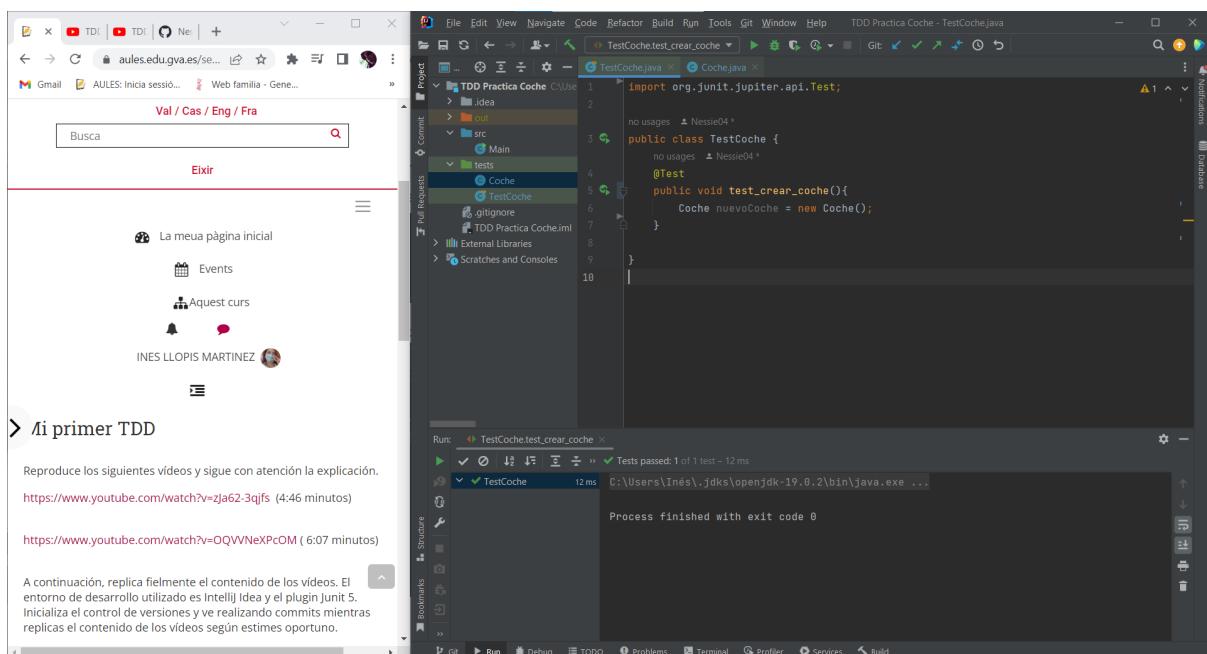
Creamos el método “test_crear_coche()”. Hacemos commit.



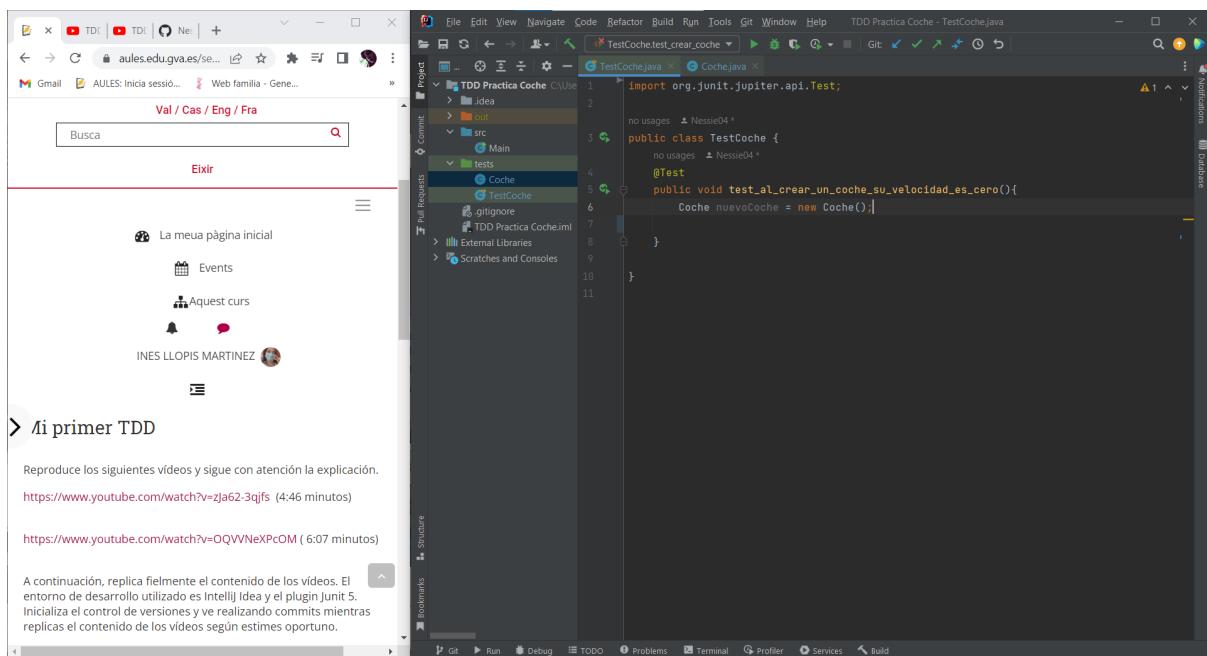
Creamos una clase “Coche” manteniendo el cursor encima de la palabra “Coche” en rojo y haciendo clic en “Crear clase Coche”. Hacemos commit.



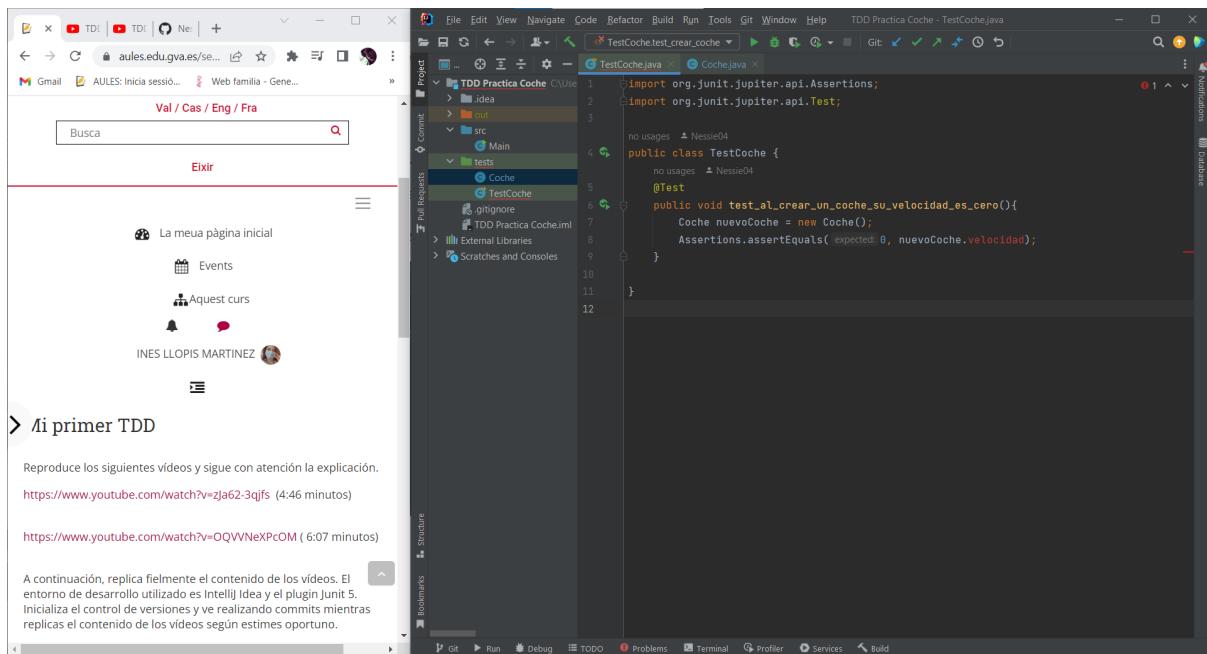
Realizamos una primera prueba para ver si el programa funciona correctamente haciendo clic en “Run”.



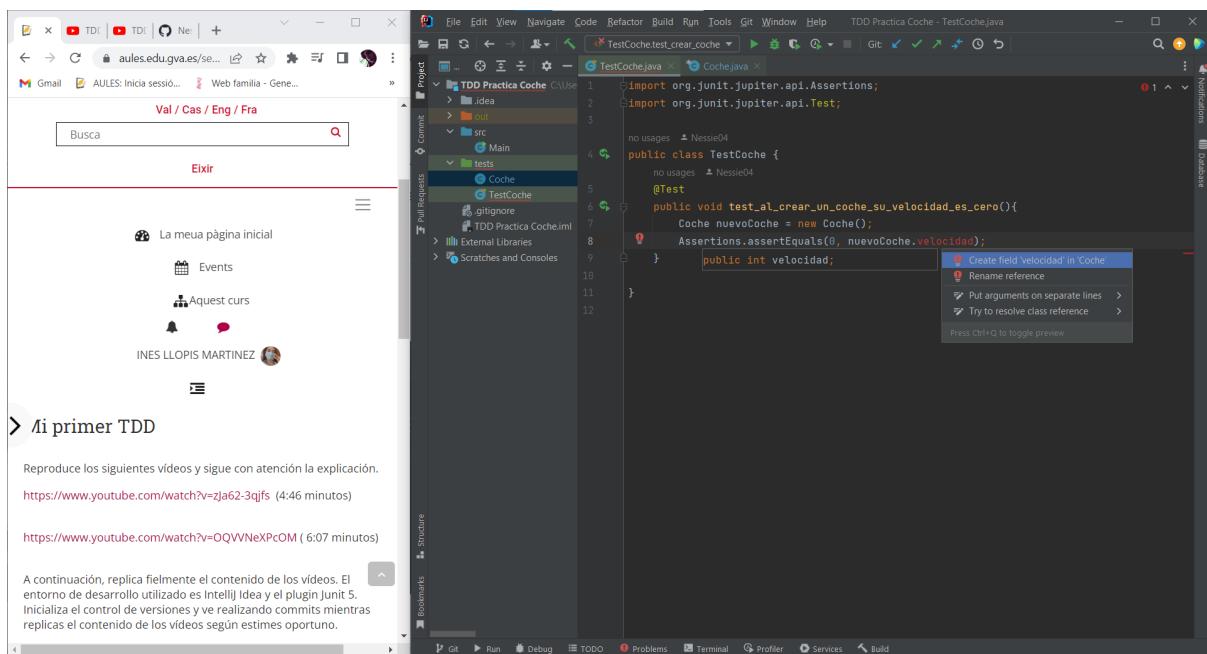
Refactorizamos el nombre del método “test_crear_coche” y lo cambiamos por “test_al_crear_un_coche_su_velocidad_es_cero” haciendo clic derecho sobre el nombre del metodo y seleccionando “Refactor > Rename”. Hacemos commit.



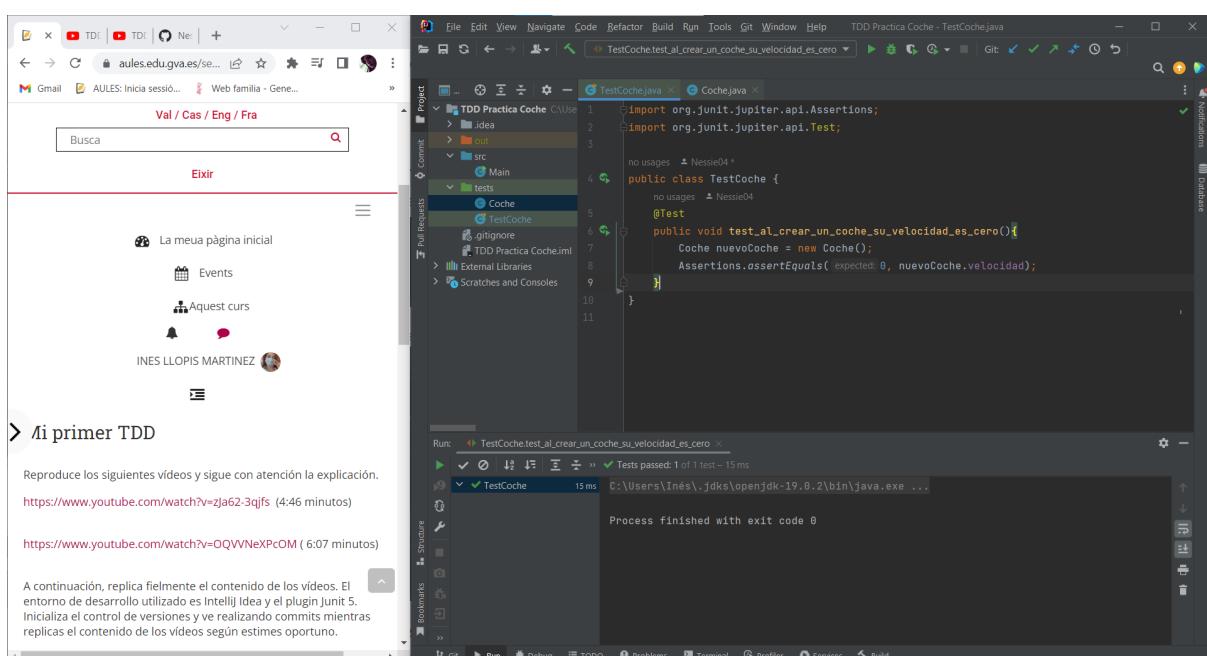
Utilizamos un assertions (y lo importamos si no lo estaba) para establecer que la velocidad esperada del coche será 0 y que el valor real será “nuevoCoche.velocidad”. Hacemos commit.



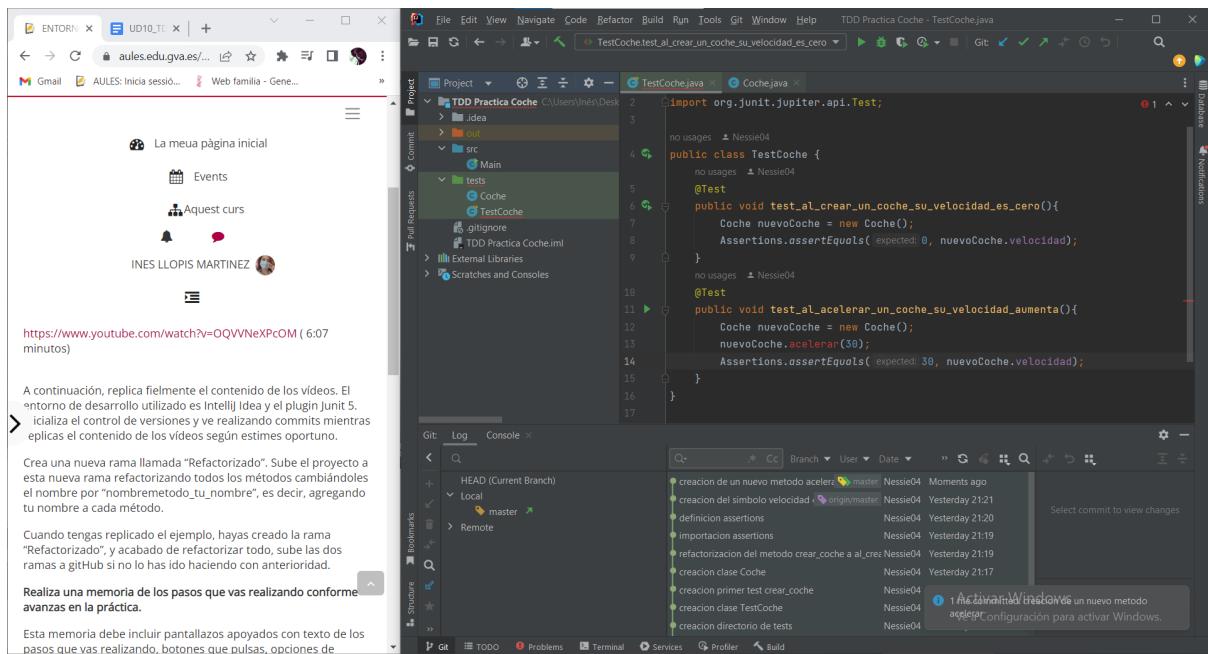
Creamos el símbolo “velocidad” dentro de la clase “Coche” manteniendo el cursor encima de la palabra “velocidad” en rojo y haciendo clic en “Crear nuevo símbolo velocidad”. Hacemos commit.



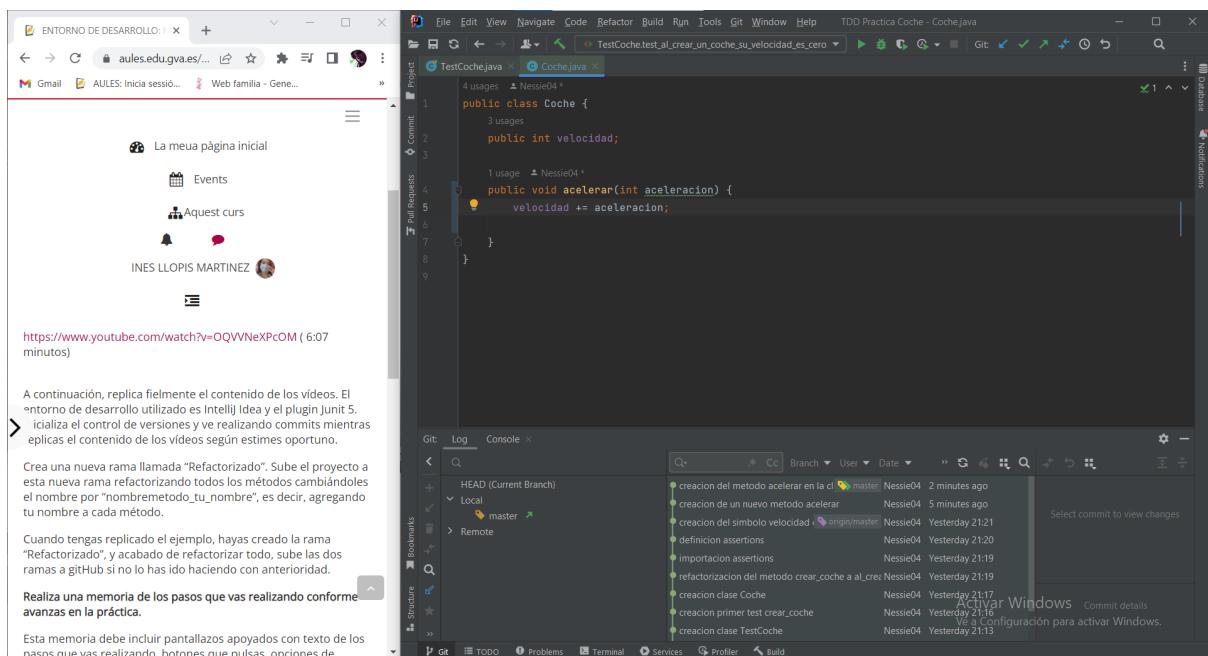
Realizamos una segunda prueba al programa para comprobar que sigue funcionando como debería.



Creamos un nuevo método llamado "test_al_acelerar_un_coche_su_velocidad_aumenta()". Hacemos commit.



Creamos el método “acelerar” dentro de la clase Coche manteniendo el cursor sobre la palabra “acelerar” en rojo y haciendo clic en “Crear nuevo método acelerar”. Hacemos commit.



Realizamos una tercera prueba al programa para comprobar que sigue funcionando como debería.

A continuación, replica fielmente el contenido de los vídeos. El entorno de desarrollo utilizado es IntelliJ Idea y el plugin Junit 5. Actualiza el control de versiones y ve realizando commits mientras ejecutas el contenido de los vídeos según estimes oportuno.

Crea una nueva rama llamada "Refactorizado". Sube el proyecto a esta nueva rama refactorizando todos los métodos cambiándoles el nombre por "nombremetodo_tu_nombre", es decir, agregando tu nombre a cada método.

Cuando tengas replicado el ejemplo, hayas creado la rama "Refactorizado", y acabado de refactorizar todo, sube las dos ramas a GitHub si no lo has ido haciendo con anterioridad.

Realiza una memoria de los pasos que vas realizando conforme avanzas en la práctica.

Esta memoria debe incluir pantallazos apoyados con texto de los pasos que vas realizando, botones que pulsas, opciones de

Creamos un nuevo método "test_al_decelerar_un_coche_su_velocidad_disminuye()". Hacemos commit.

A continuación, replica fielmente el contenido de los vídeos. El entorno de desarrollo utilizado es IntelliJ Idea y el plugin Junit 5. Actualiza el control de versiones y ve realizando commits mientras ejecutas el contenido de los vídeos según estimes oportuno.

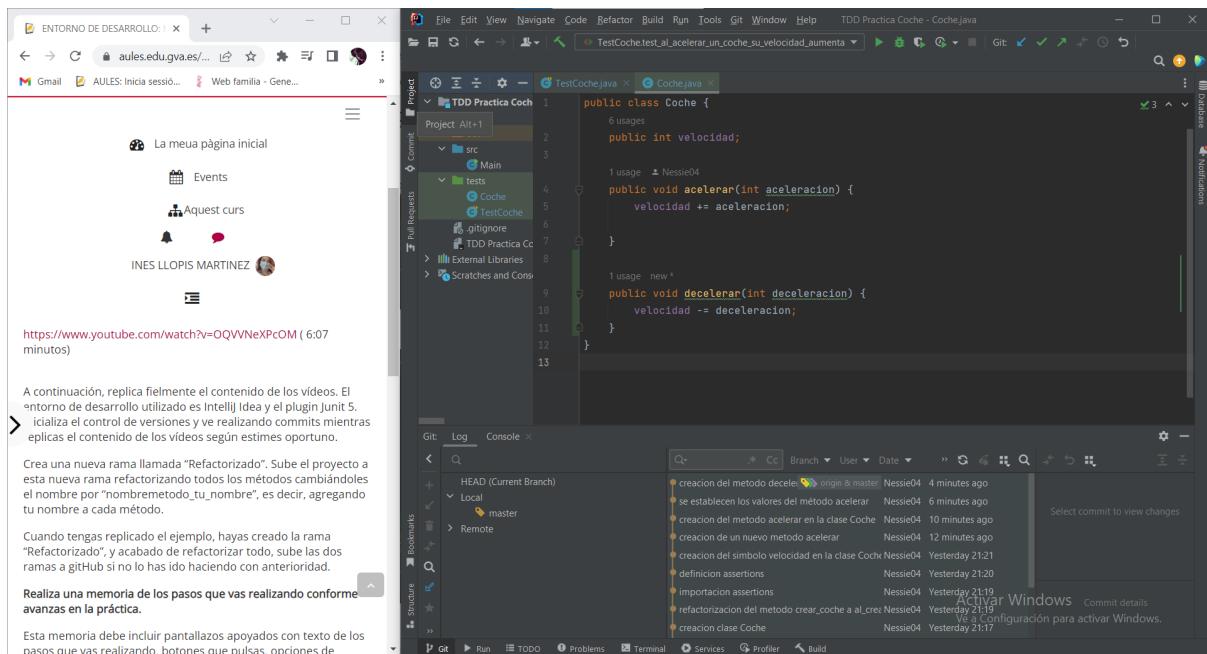
Crea una nueva rama llamada "Refactorizado". Sube el proyecto a esta nueva rama refactorizando todos los métodos cambiándoles el nombre por "nombremetodo_tu_nombre", es decir, agregando tu nombre a cada método.

Cuando tengas replicado el ejemplo, hayas creado la rama "Refactorizado", y acabado de refactorizar todo, sube las dos ramas a GitHub si no lo has ido haciendo con anterioridad.

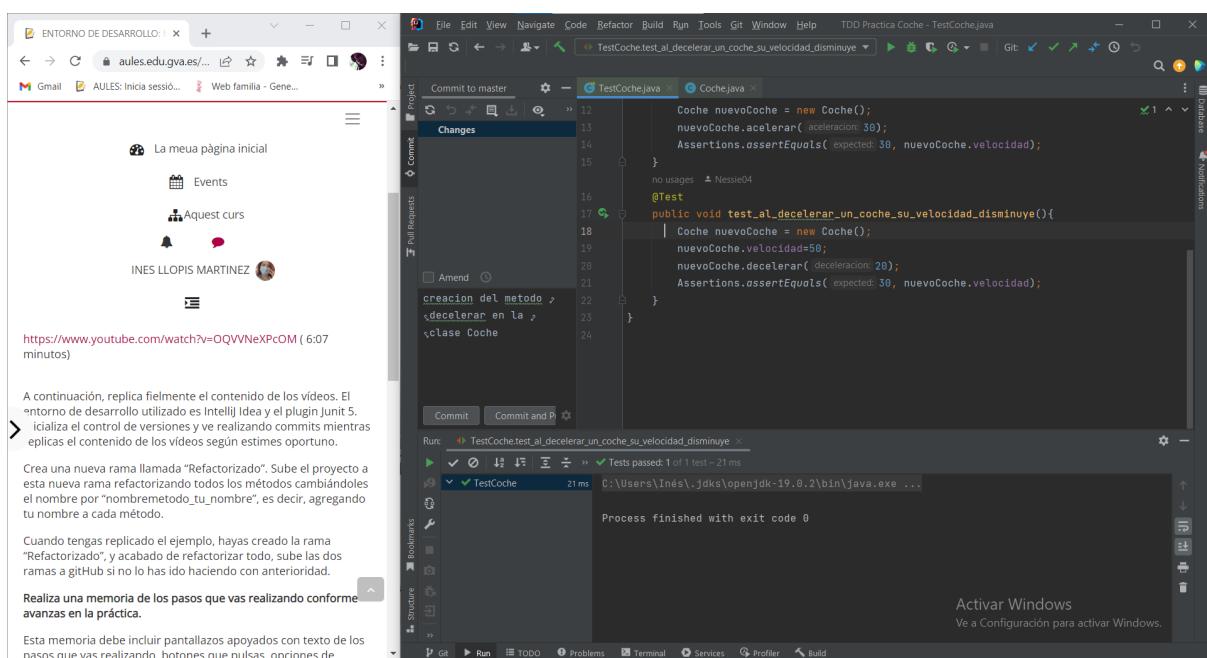
Realiza una memoria de los pasos que vas realizando conforme avanzas en la práctica.

Esta memoria debe incluir pantallazos apoyados con texto de los pasos que vas realizando, botones que pulsas, opciones de

Creamos el método "decelerar" dentro de la clase Coche manteniendo el cursor sobre la palabra "decelerar" en rojo y haciendo clic en "Crear nuevo método decelerar". Hacemos commit.



Realizamos una cuarta prueba al programa para comprobar que sigue funcionando como debería.



Creamos el método
"test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero".
Hacemos commit.

```

    Assertions.assertEquals(expected: 30, nuevoCoche.velocidad);
    no usages ▾ Nessie04
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_disminue(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad=50;
        nuevoCoche.decelerar( deceleracion: 20);
        Assertions.assertEquals(expected: 30, nuevoCoche.velocidad);
    }
    no usages ▾ Nessie04
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad=50;
        nuevoCoche.decelerar( deceleracion: 80);
        Assertions.assertEquals(expected: 0, nuevoCoche.velocidad);
    }
}

```

Activar Windows
Ve a Configuración para activar Windows.

Realizamos una quinta prueba al programa para comprobar que sigue funcionando como debería y vemos que no se ha superado la prueba porque el resultado obtenido y el esperado no son iguales.

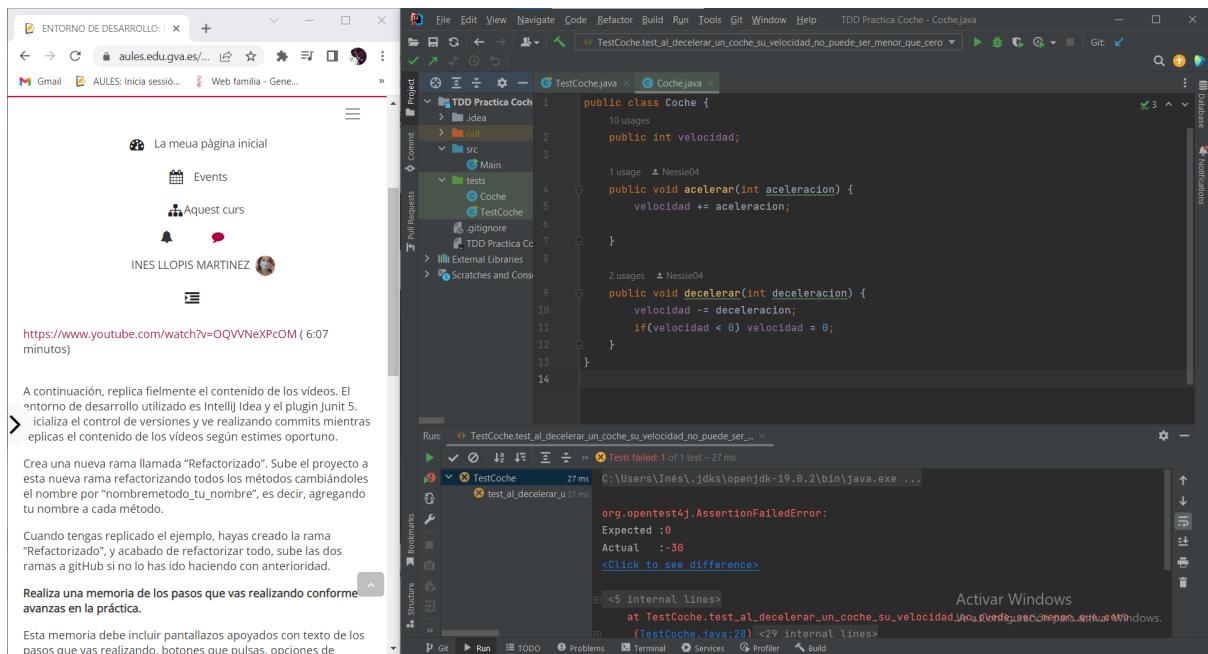
```

    Assertions.assertEquals(expected: 30, nuevoCoche.velocidad);
    no usages ▾ Nessie04
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_disminue(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad=50;
        nuevoCoche.decelerar( deceleracion: 20);
        Assertions.assertEquals(expected: 30, nuevoCoche.velocidad);
    }
    no usages ▾ Nessie04
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad=50;
        nuevoCoche.decelerar( deceleracion: 80);
        Assertions.assertEquals(expected: 0, nuevoCoche.velocidad);
    }
}

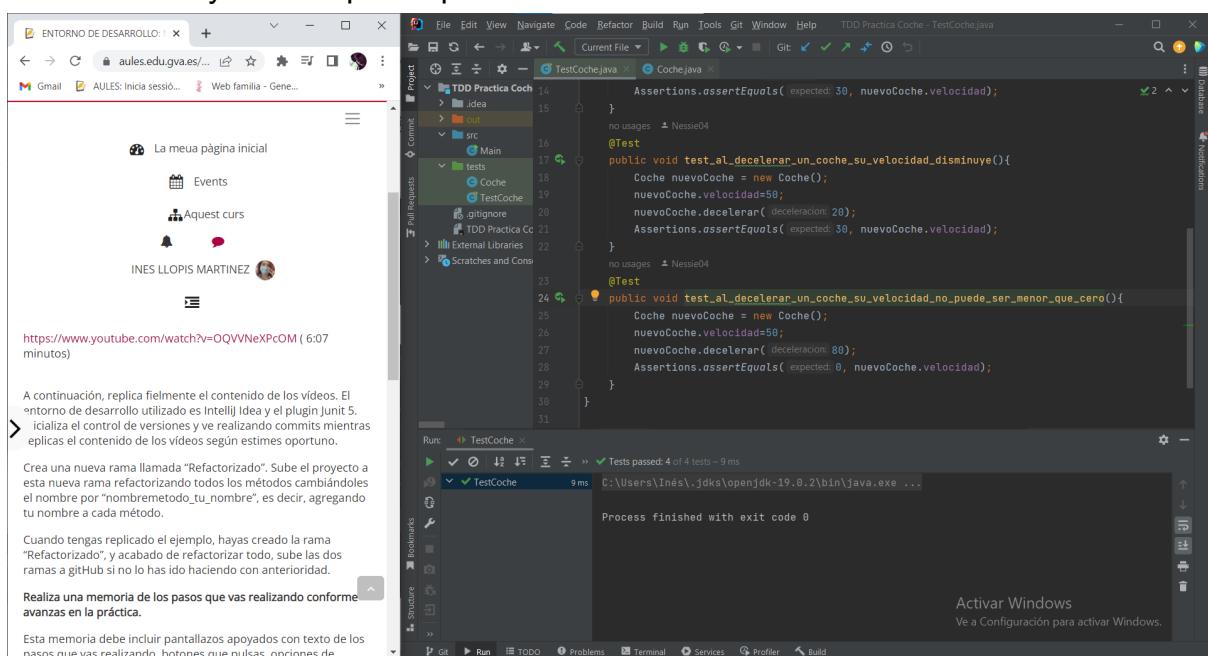
```

Activar Windows
Ve a Configuración para activar Windows.

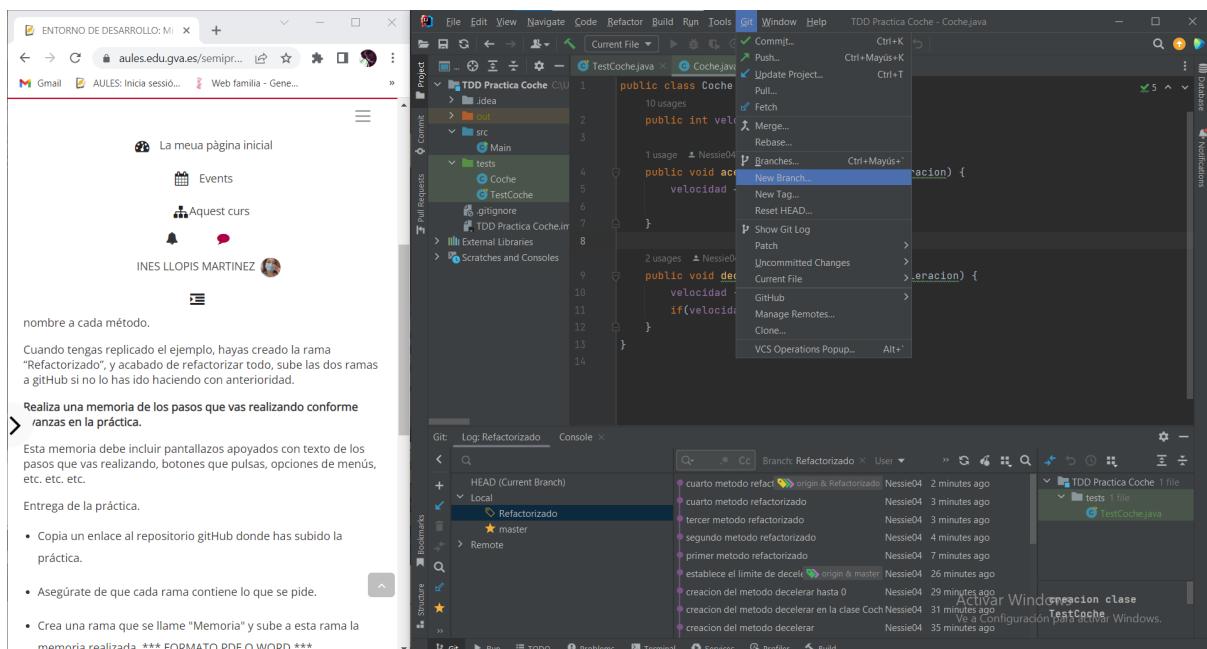
Establecemos el límite de deceleración en 0 mediante un *if* en el método “decelerar” de la clase Coche. Hacemos commit.



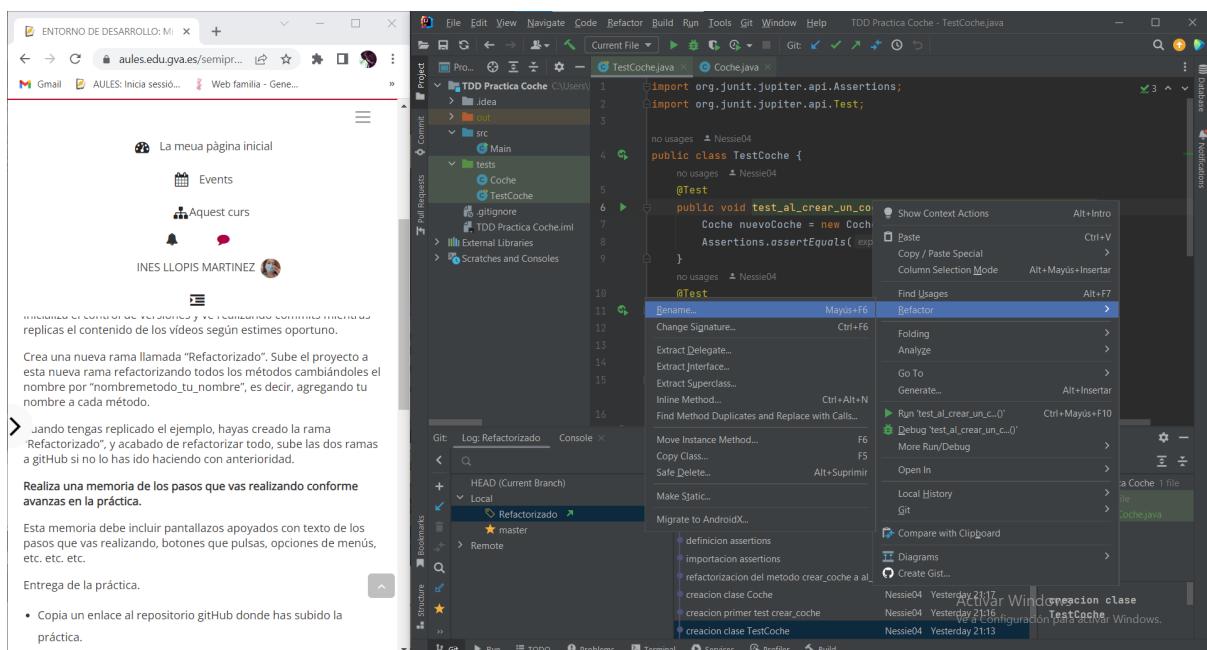
Realizamos una sexta prueba al programa para comprobar que sigue funcionando como debería y vemos que se pasan los 4 tests correctamente.



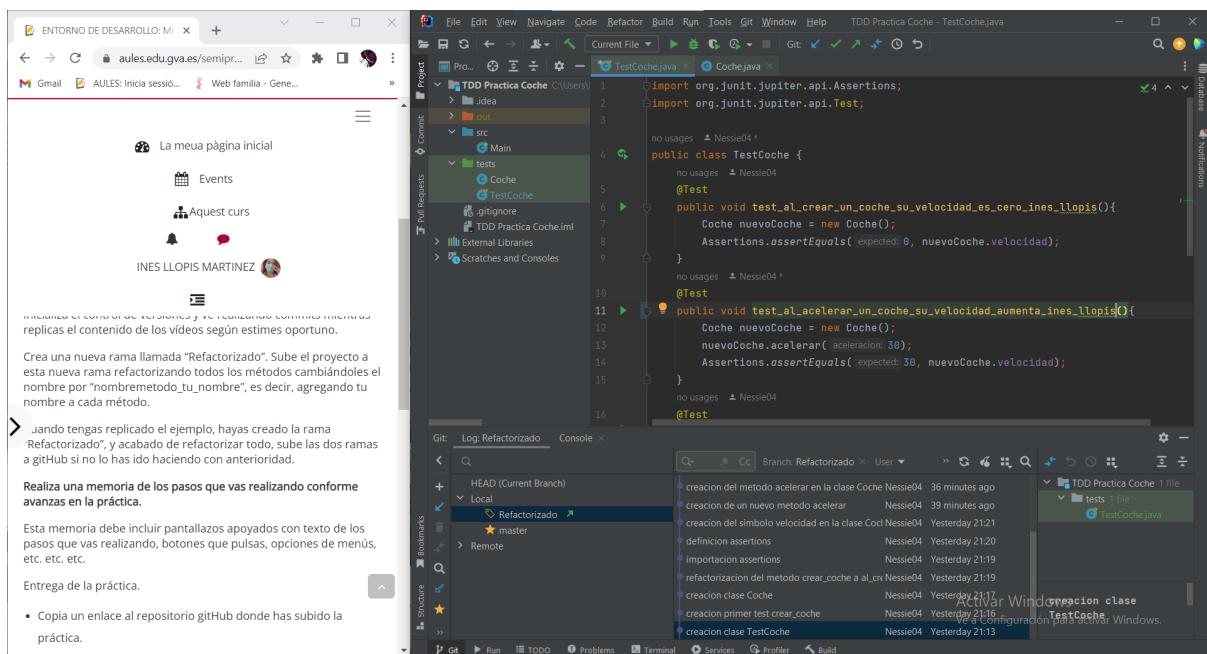
Creamos una nueva rama llamada “Refactorizado” con la opción “New Branch...” del menú Git.



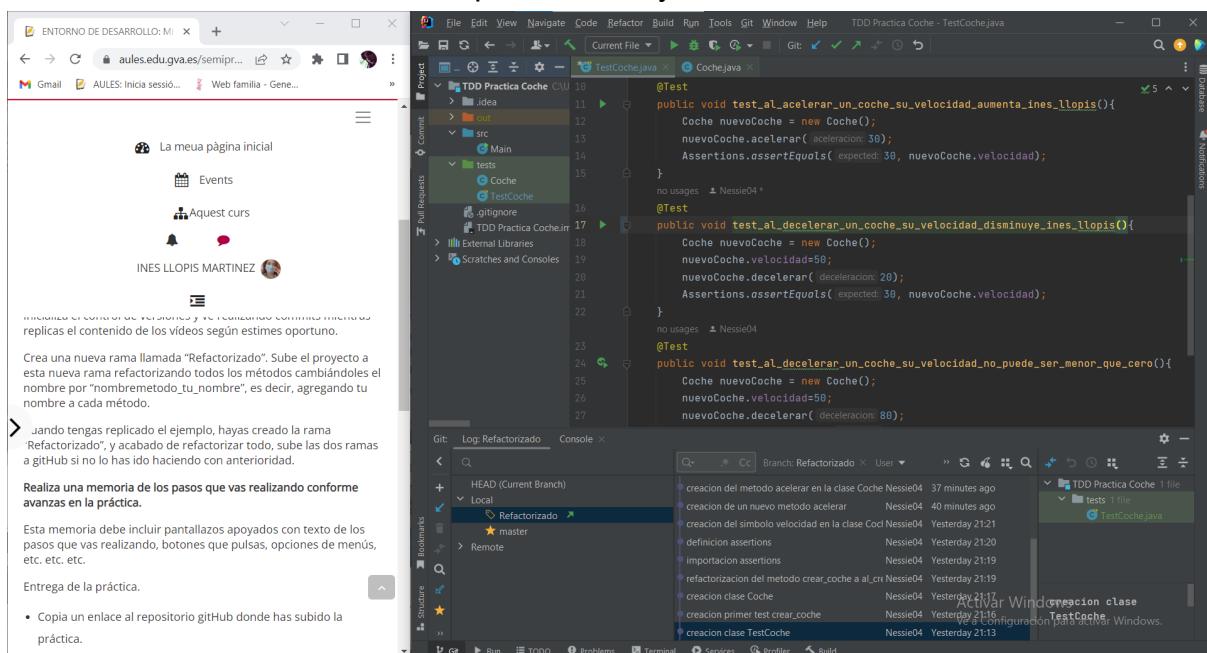
Refactorizamos el primer método para añadir nuestro nombre al final del mismo haciendo clic derecho sobre el nombre del método y seleccionando “Refactor > Rename...”.



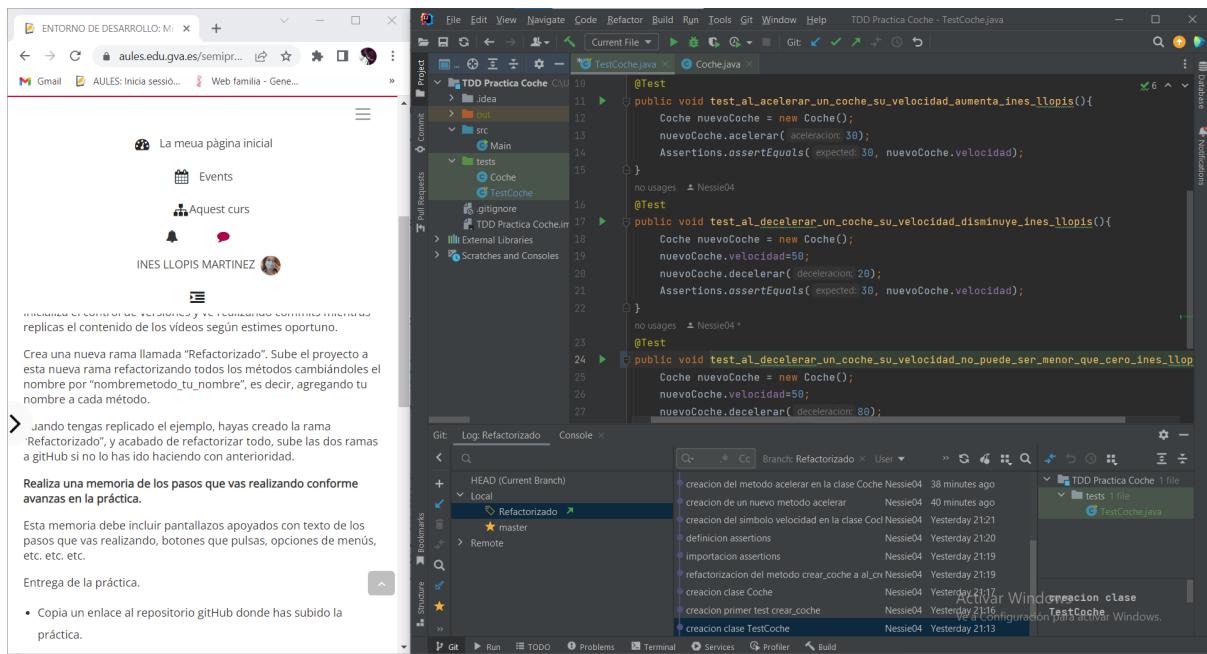
Refactorizamos el segundo método pulsando Mayús+F6.



Refactorizamos el tercer método pulsando Mayús+F6.



Refactorizamos el cuarto método pulsando Mayús+F6.



Refactorizamos los dos métodos de la clase Coche pulsando Mayús+F6.

