

# UD 10:

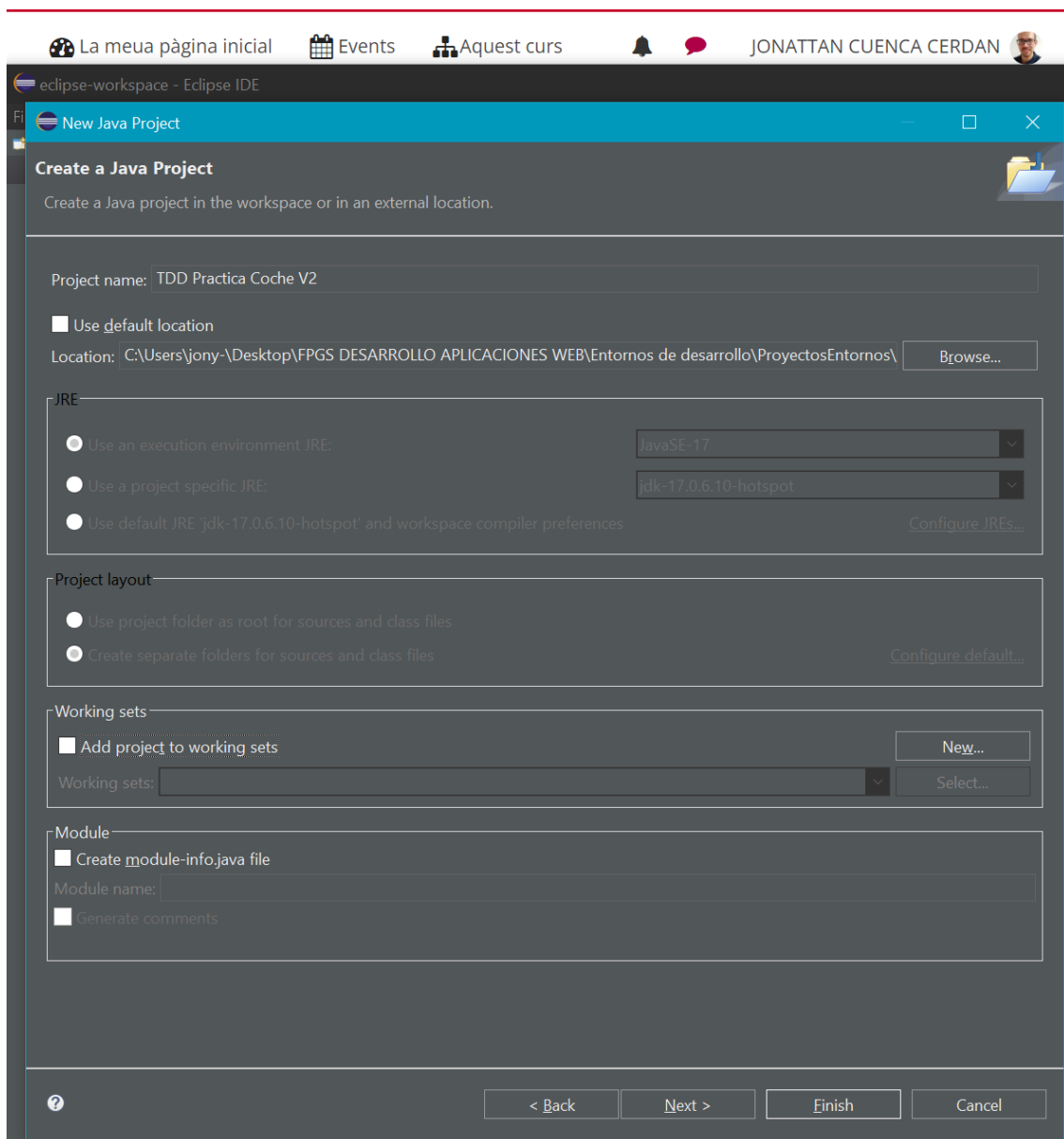
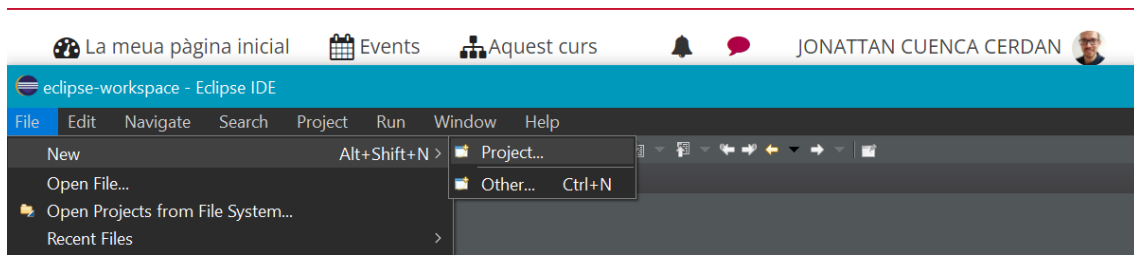
# Práctica

# Mi primer

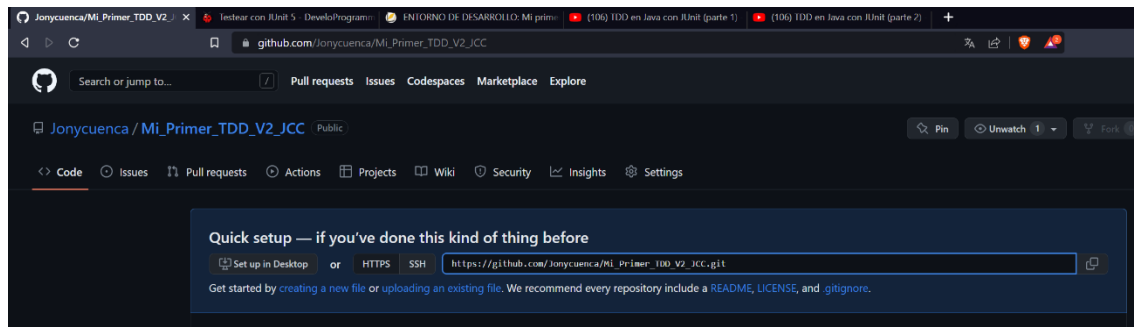
# TDD con

# Eclipse

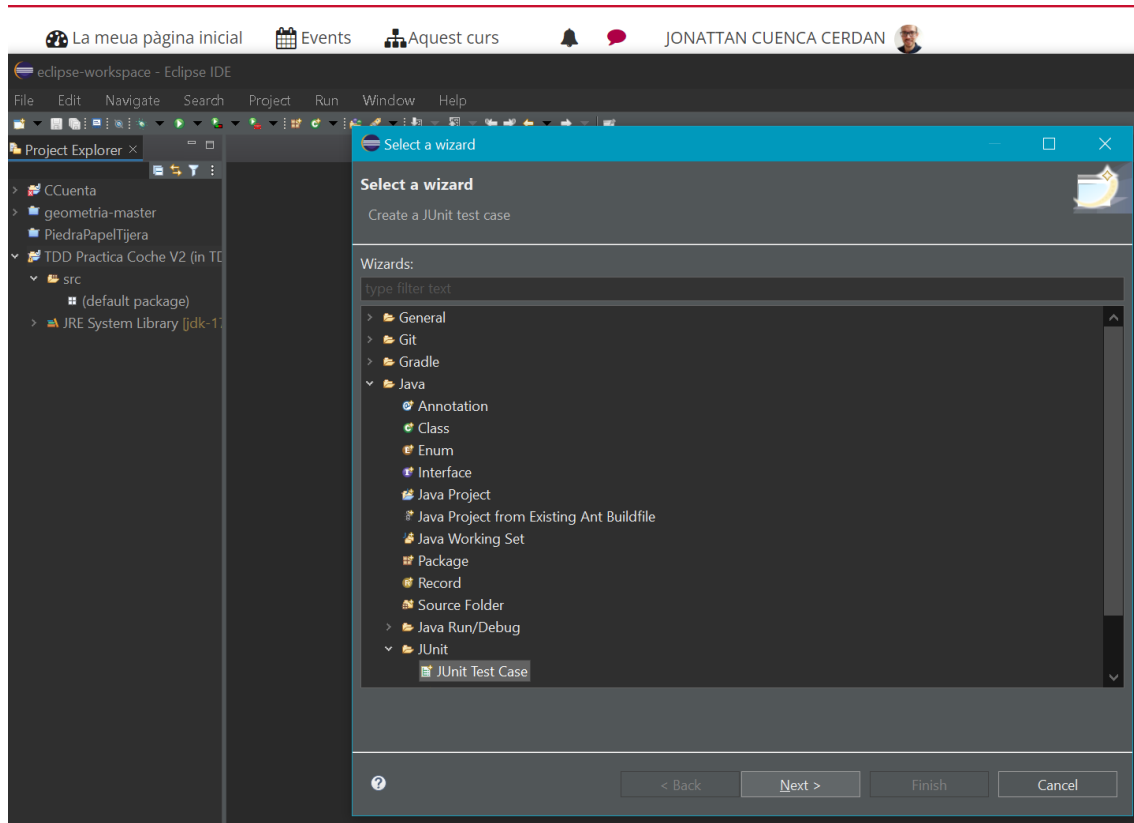
Vamos a poner en práctica el TDD en un proyecto de java utilizando el IDE Eclipse. Para esto, creamos un nuevo proyecto, seleccionamos un Proyecto Java y lo nombramos TDD Practica Coche V2.



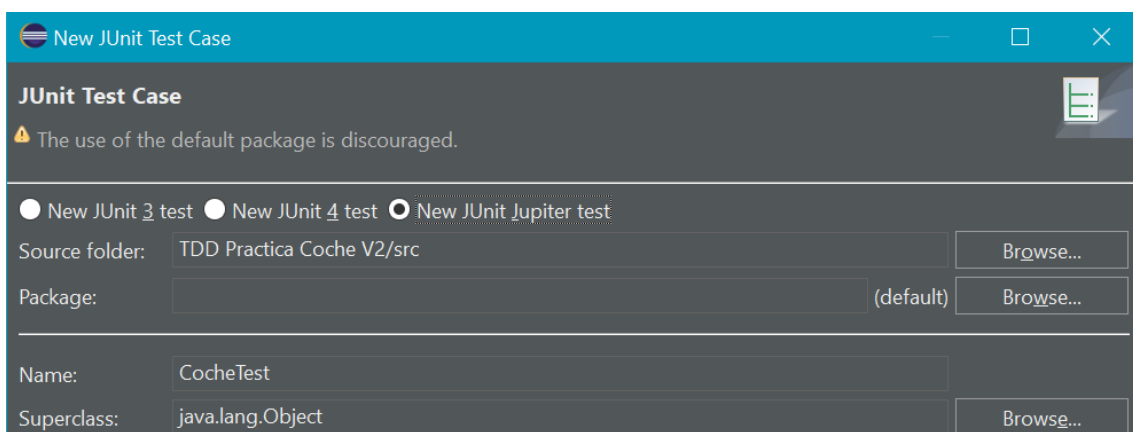
Creamos también un nuevo repositorio remoto en GitHub y copiamos la url del directorio para sincronizar con nuestro proyecto en el IDE Eclipse.



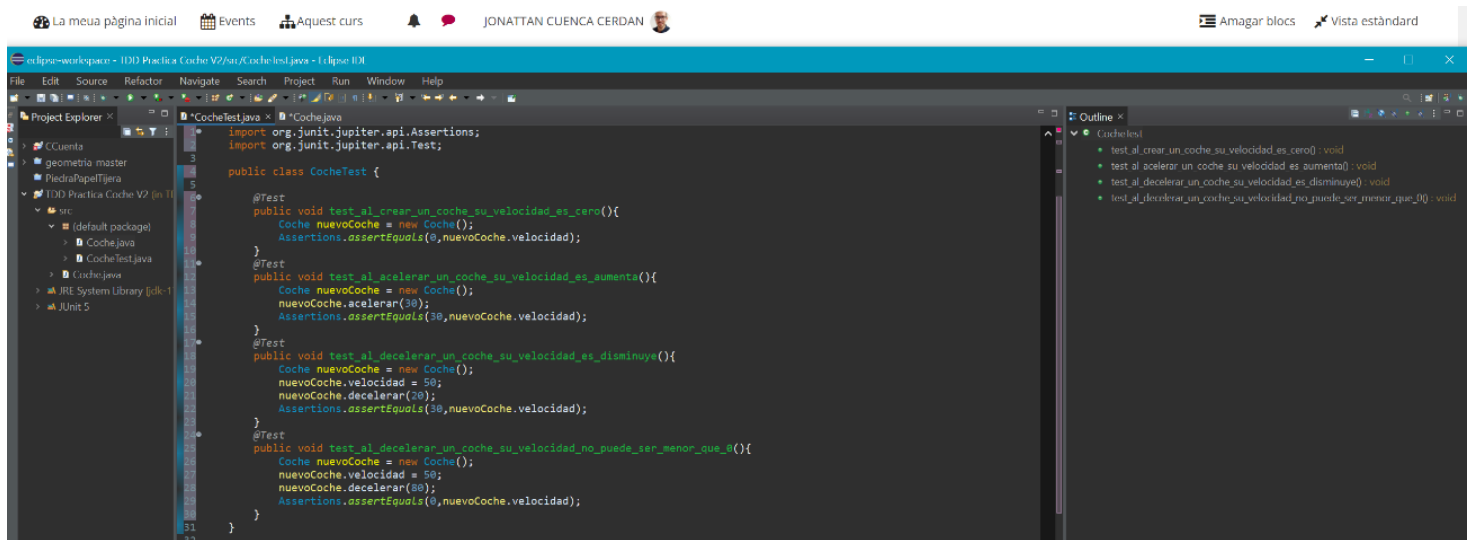
Comenzamos a crear nuestro proyecto para realizar los test. Para esto no situamos sobre el proyecto, mostramos el menú contextual y seleccionamos: *New -> Java -> JUnit -> JUnit Test Case -> Next*



Seleccionamos New JUnit Jupiter test y nombramos la clase para el test como CocheTest.



A continuación. Copiamos introducimos el código, tanto métodos como la clase Coche, como hicimos en el ejercicio anterior con IntelliJ siguiendo los pasos del vídeo. El código quedaría de la siguiente manera:



The screenshot shows the Eclipse IDE interface. The main editor displays `CocheTest.java` with the following code:

```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class CocheTest {

    @Test
    public void test_al_crear_un_coche_su_velocidad_es_cero(){
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0,nuevoCoche.velocidad);
    }

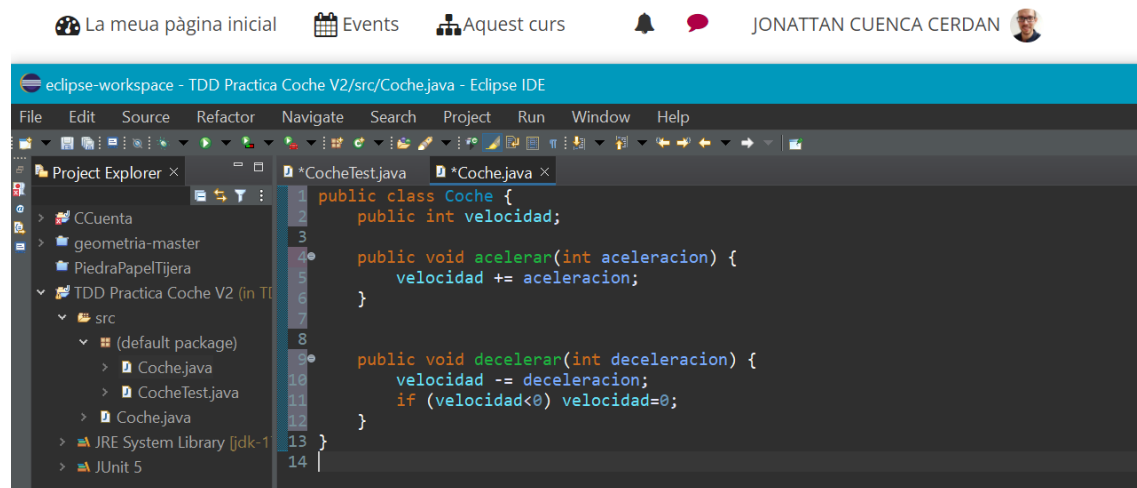
    @Test
    public void test_al_acelerar_un_coche_su_velocidad_es_aumenta(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.acelerar(30);
        Assertions.assertEquals(30,nuevoCoche.velocidad);
    }

    @Test
    public void test_al_decelerar_un_coche_su_velocidad_es_disminuye(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad = 50;
        nuevoCoche.decelerar(20);
        Assertions.assertEquals(30,nuevoCoche.velocidad);
    }

    @Test
    public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_0(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad = 50;
        nuevoCoche.decelerar(80);
        Assertions.assertEquals(0,nuevoCoche.velocidad);
    }
}
```

The Outline view on the right shows the test methods:

- test\_al\_crear\_un\_coche\_su\_velocidad\_es\_cero() : void
- test\_al\_acelerar\_un\_coche\_su\_velocidad\_es\_aumenta() : void
- test\_al\_decelerar\_un\_coche\_su\_velocidad\_es\_disminuye() : void
- test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_ser\_menor\_que\_0() : void



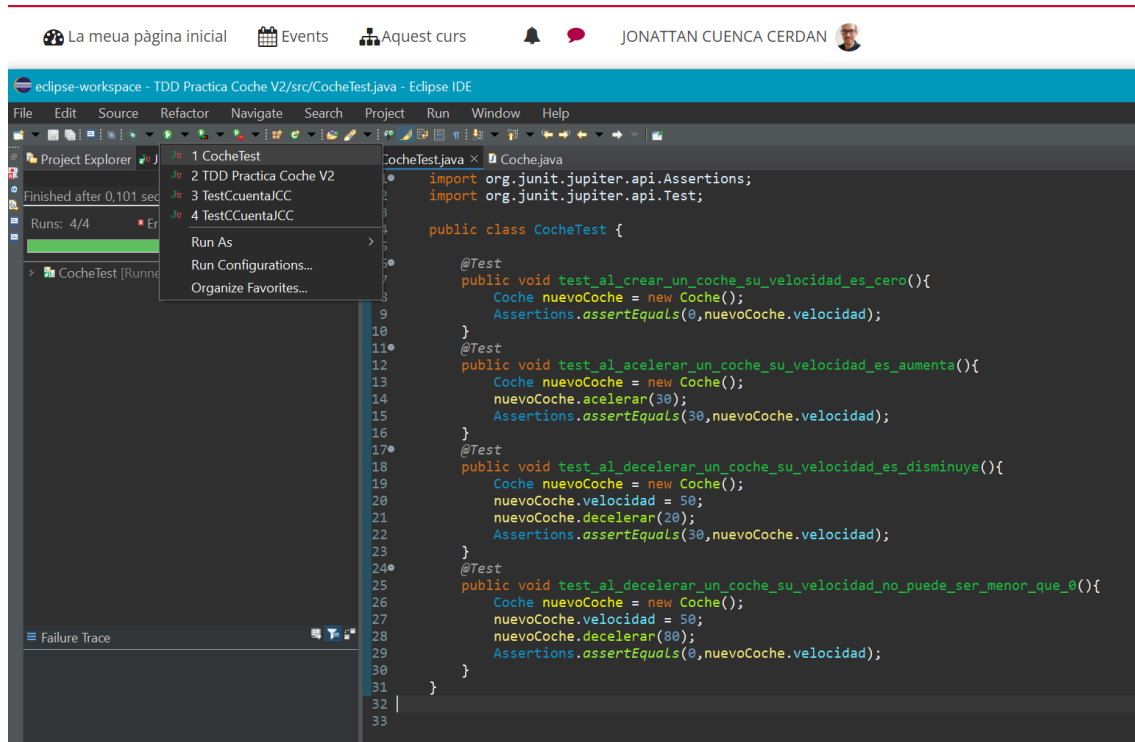
The screenshot shows the Eclipse IDE interface. The main editor displays `Coche.java` with the following code:

```
public class Coche {
    public int velocidad;

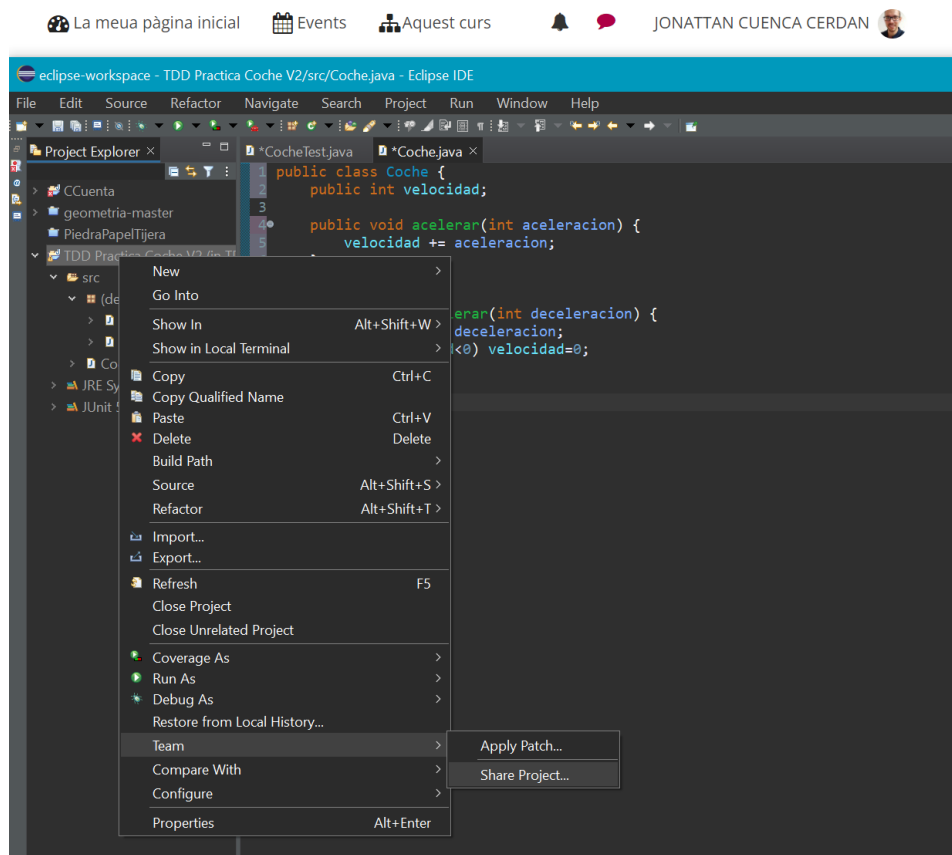
    public void acelerar(int aceleracion) {
        velocidad += aceleracion;
    }

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

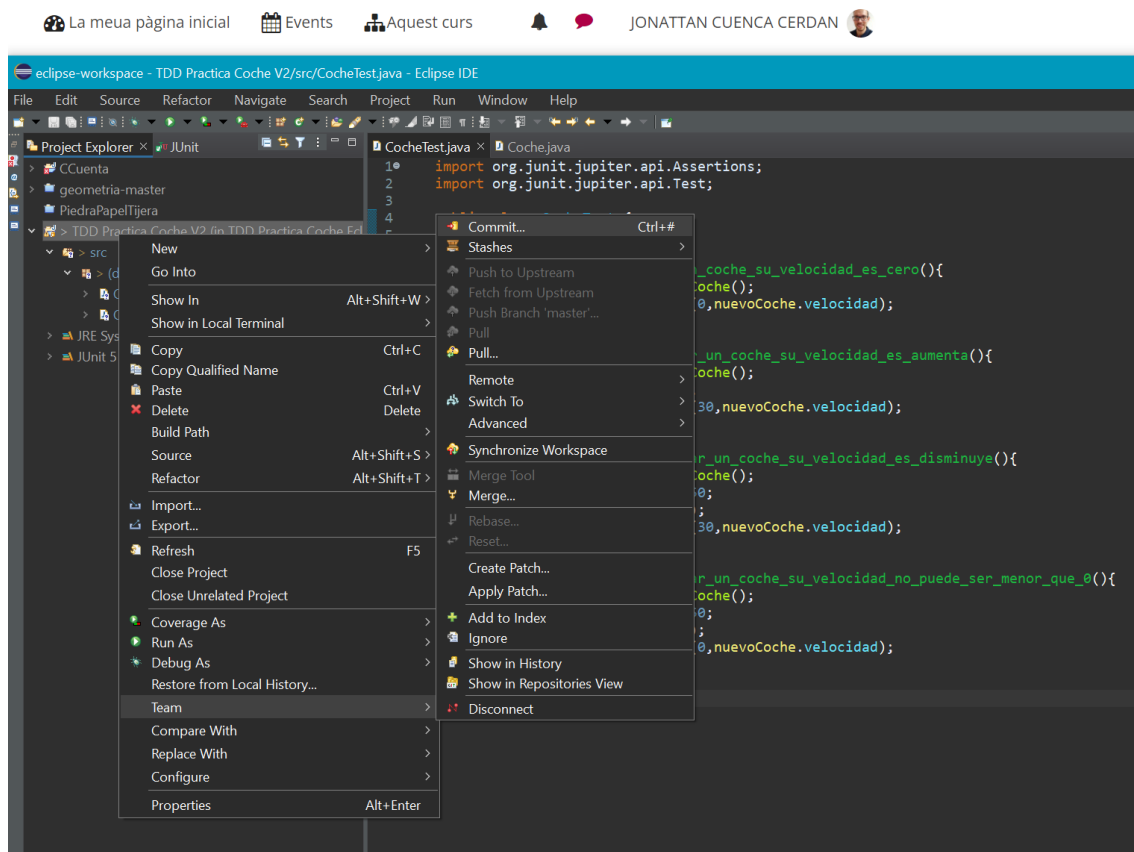
Una vez hemos introducido el código. Realizamos una prueba para comprobar que el código funciona correctamente.



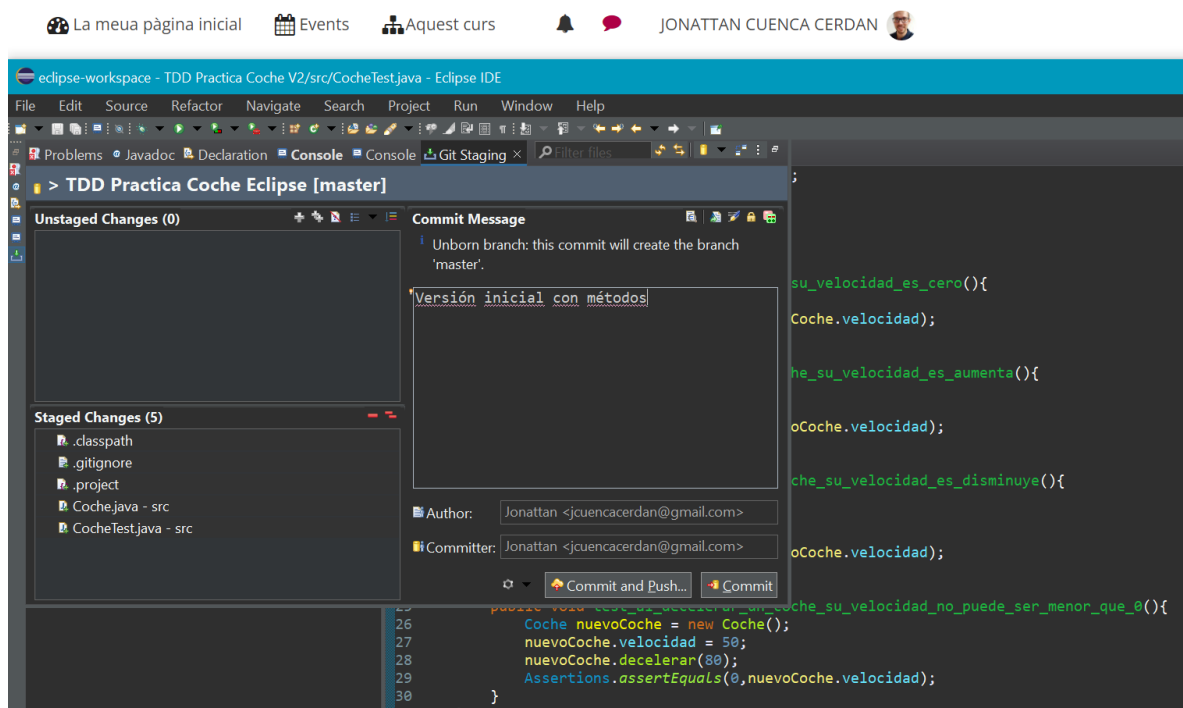
Una vez comprobado que las pruebas funcionan correctamente, creamos un repositorio de Git haciendo clic con el botón derecho del ratón sobre el nombre del proyecto, seleccionamos la opción Team -> Share Project y en la nueva pestaña que se abre seleccionamos la opción “Use or create repository in parent folder or Project”. Se creará un directorio Git en la carpeta del proyecto.



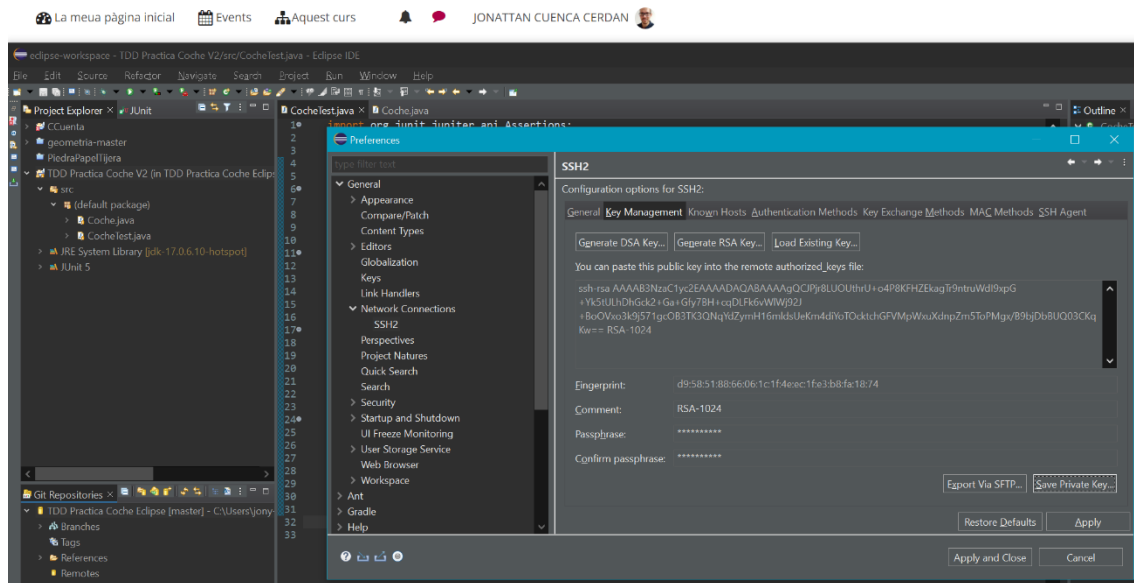
Realizaremos un commit haciendo clic derecho del ratón sobre el proyecto -> Team -> Commit



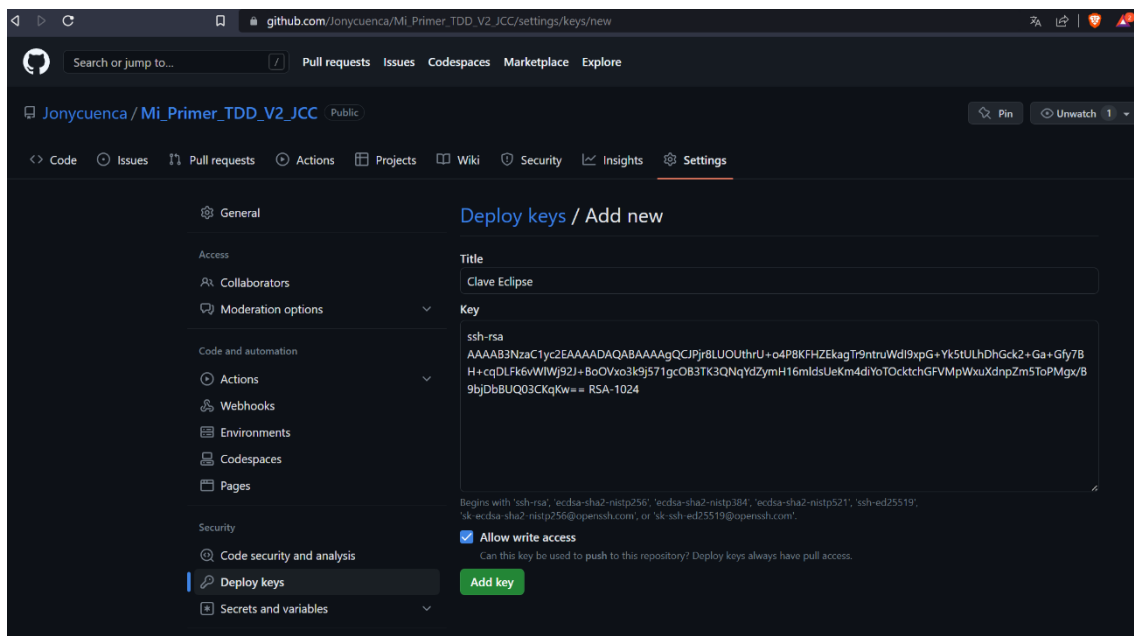
Añadimos todos los archivos y comentamos para realizar el commit.



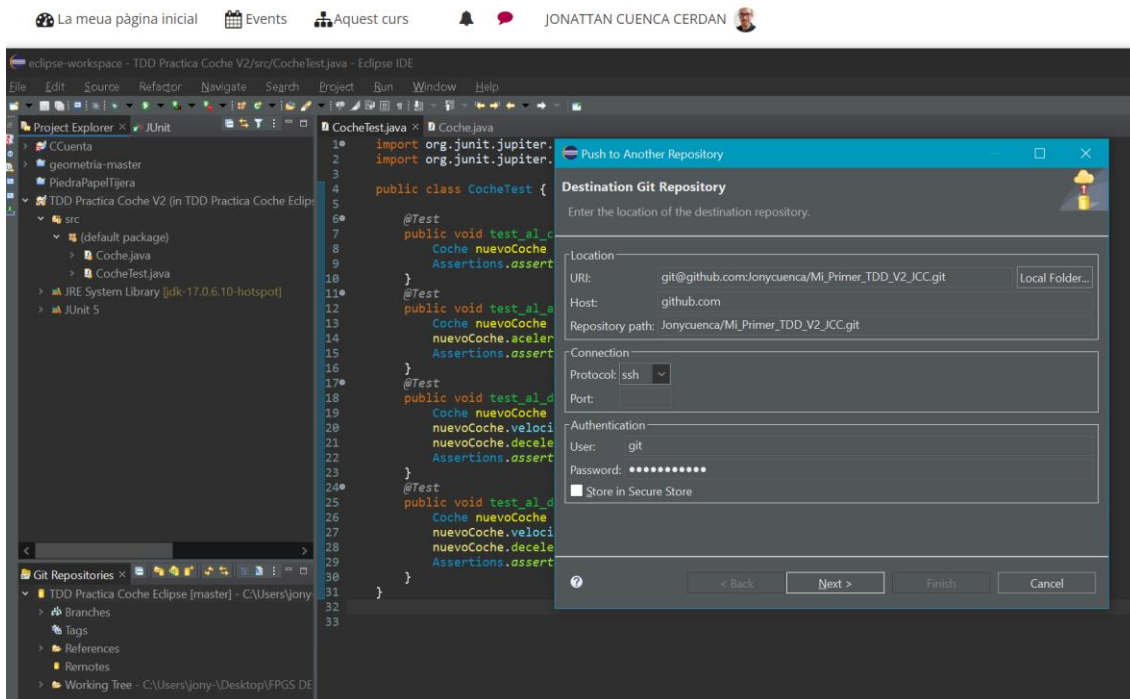
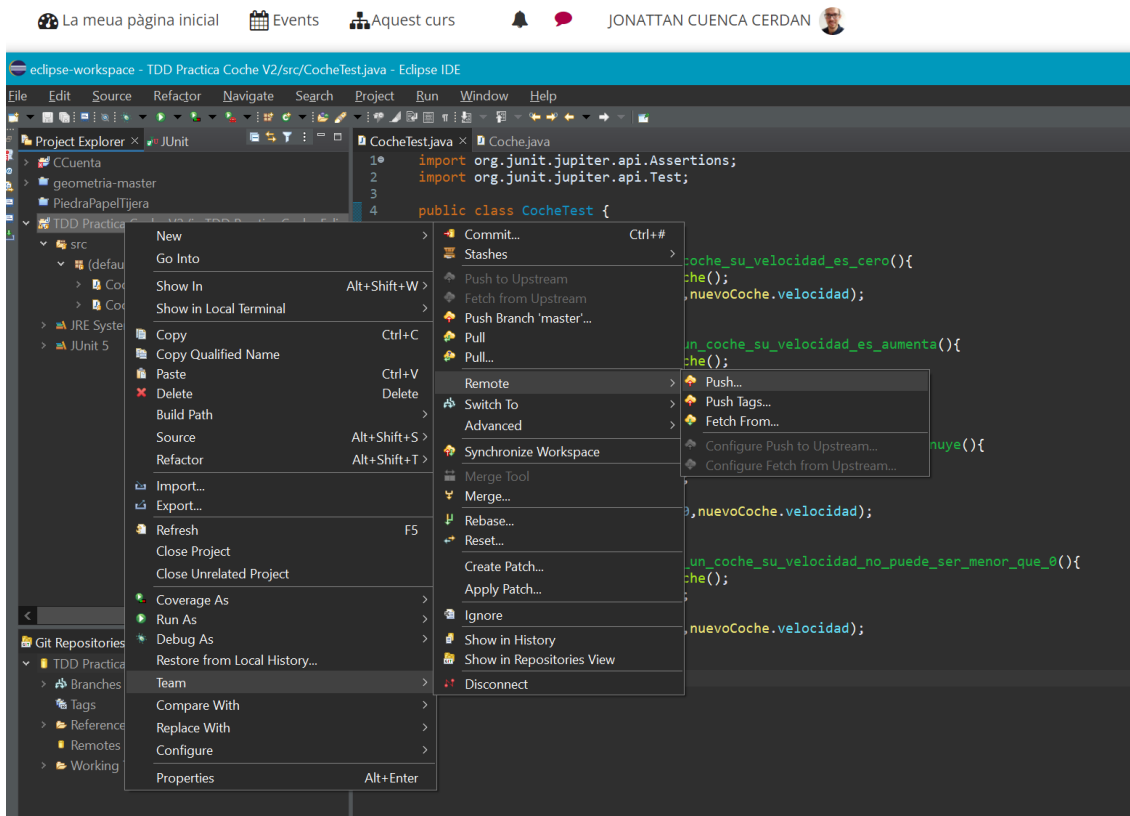
Una vez hemos realizado el commit, vamos a subirlo al repositorio remoto. Para esto, accedemos, en la barra de tareas, a Window/Preferences y en el apartado de Network Connectios -> SSH2 generamos una nueva Clave RSA. Copiamos la clave pública y aplicamos los cambios. También podemos guardar en un archivo la clave.



Una vez tenemos la clave copiada, vamos a GitHub y accedemos a Settings/Deploy keys y añadimos una clave nueva. Le ponemos un nombre y pegamos la clave. Importante seleccionar la opción de permitir la escritura.

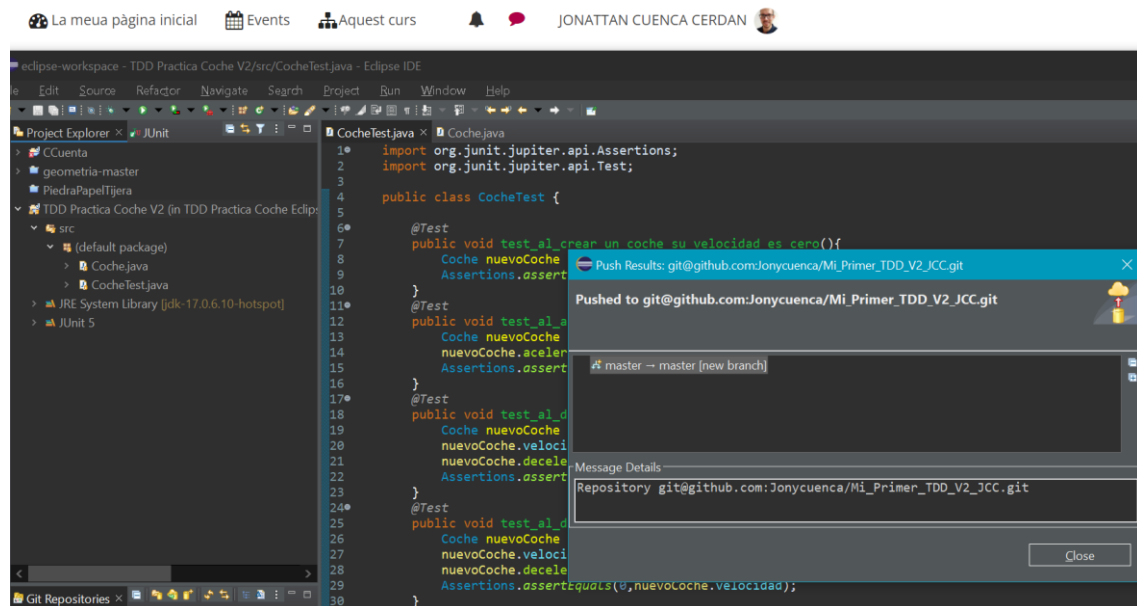


Una vez hemos realizado esto, en Eclipse, hacemos clic derecho del ratón sobre el proyecto Team -> Remote -> Push . Se nos abre una ventana nueva donde debemos introducir la url del directorio destino de GitHub.

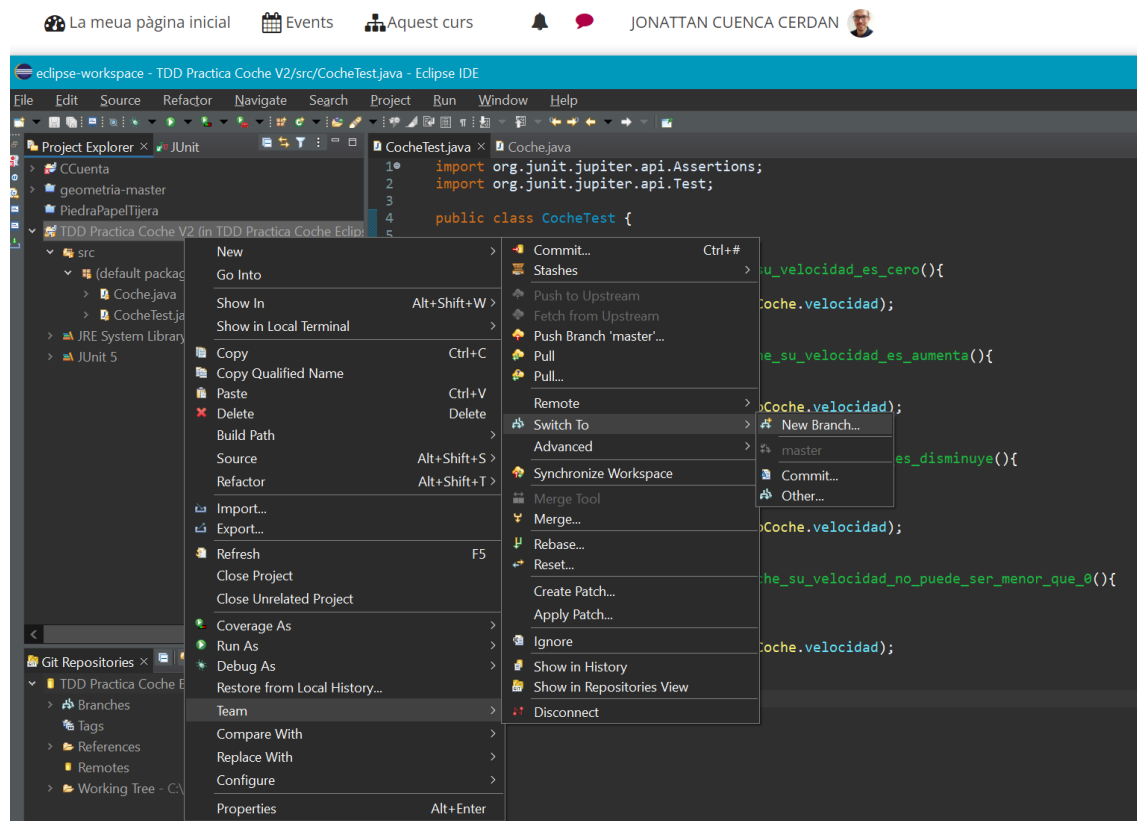




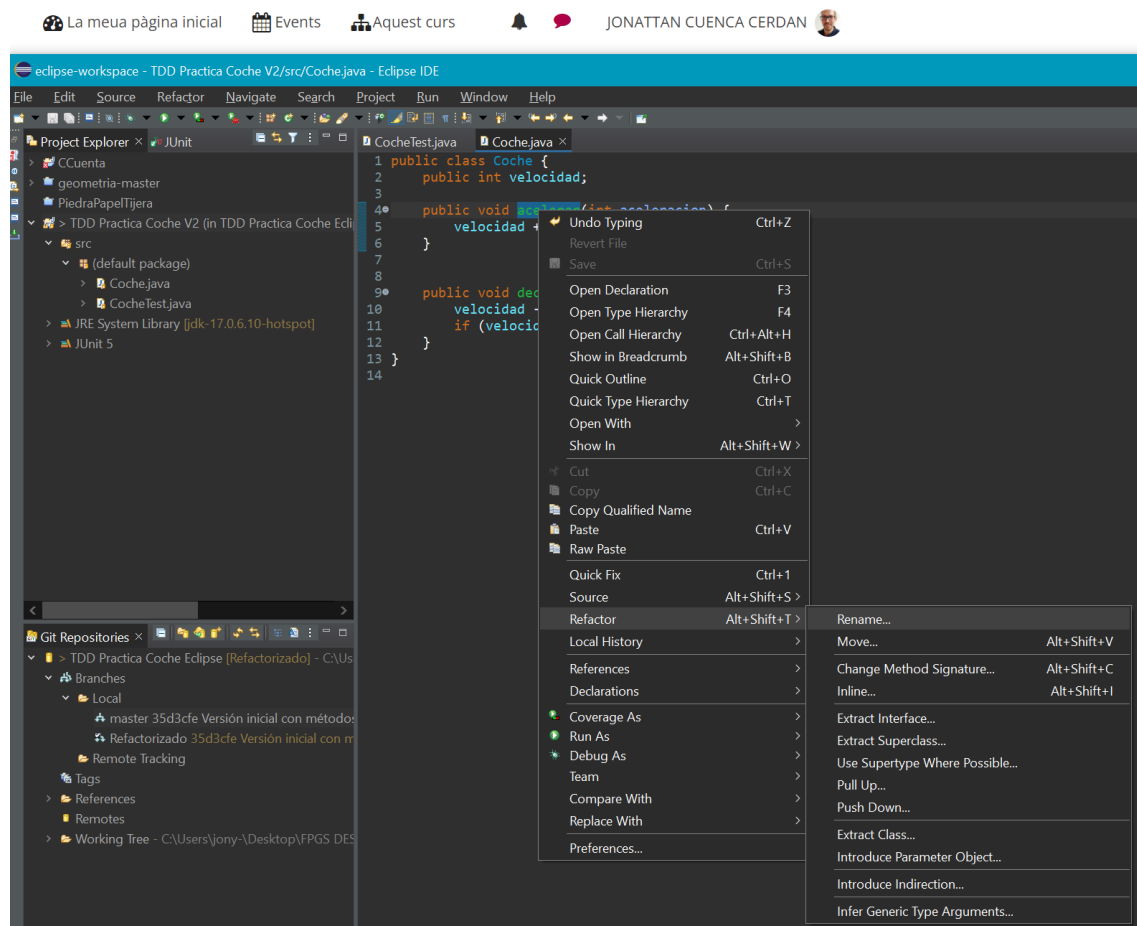
Una vez ha finalizado el proceso, el proyecto con la rama master se ha subido al repositorio de GitHub.



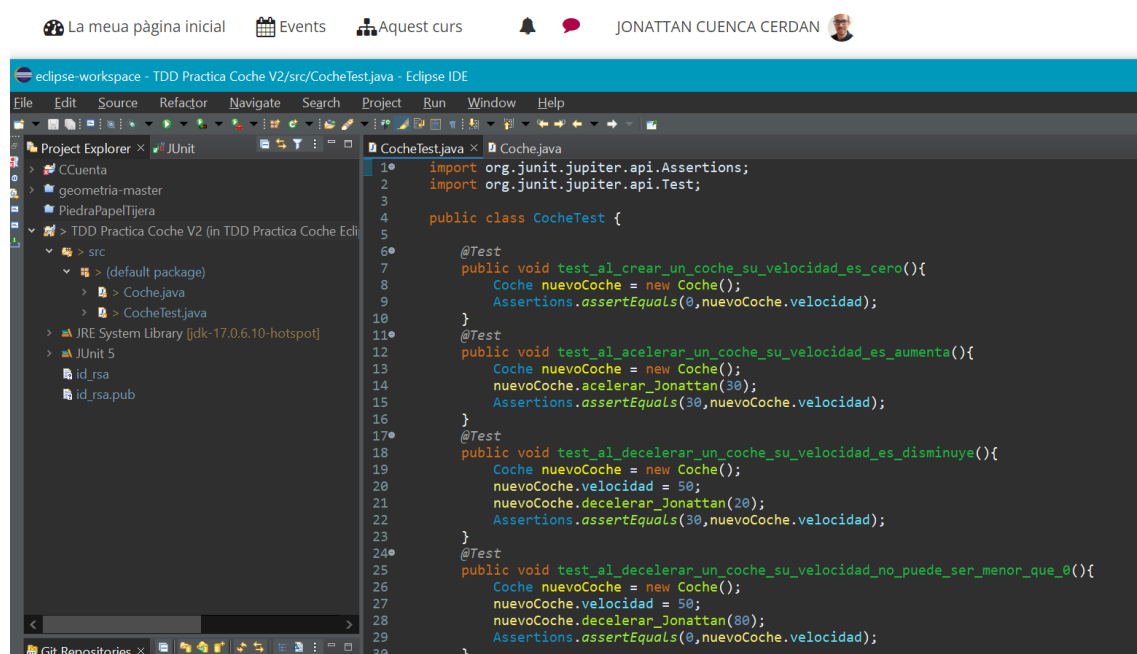
Continuamos entonces realizando la refactorización de los métodos. Les cambiaremos el nombre añadiendo nuestro nombre al final. Para esto, creamos una nueva rama y la nombramos como Refactorizado. Para hacer esto, hacemos clic derecho sobre el proyecto, Team -> Switch To -> New Branch



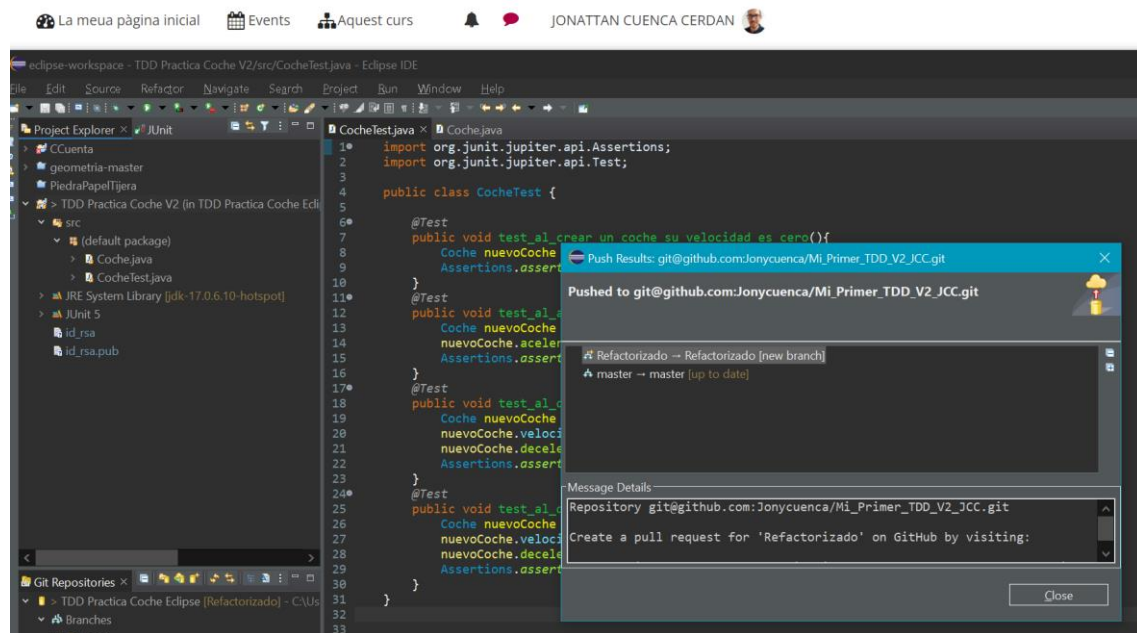
Una vez estamos en la nueva rama Refactorizado, hacemos clic derecho del ratón sobre el nombre del método que queremos renombrar y seleccionamos Refactor -> Rename . Hacemos esto con los dos métodos.



Vemos como automáticamente se sustituye el nombre en todas las líneas donde se han usado.



Tras esto, realizamos un nuevo commit y y lo subimos al directorio remoto de Github como hicimos previamente, seleccionando ahora la rama Refactorizado.



Tras esto, creamos una nueva rama “Memoria” y guardamos el archivo PDF de la memoria dentro del directorio del proyecto. Realizamos un nuevo commit y y lo subimos al directorio remoto de Github como hicimos previamente, seleccionando ahora la rama Memoria.

