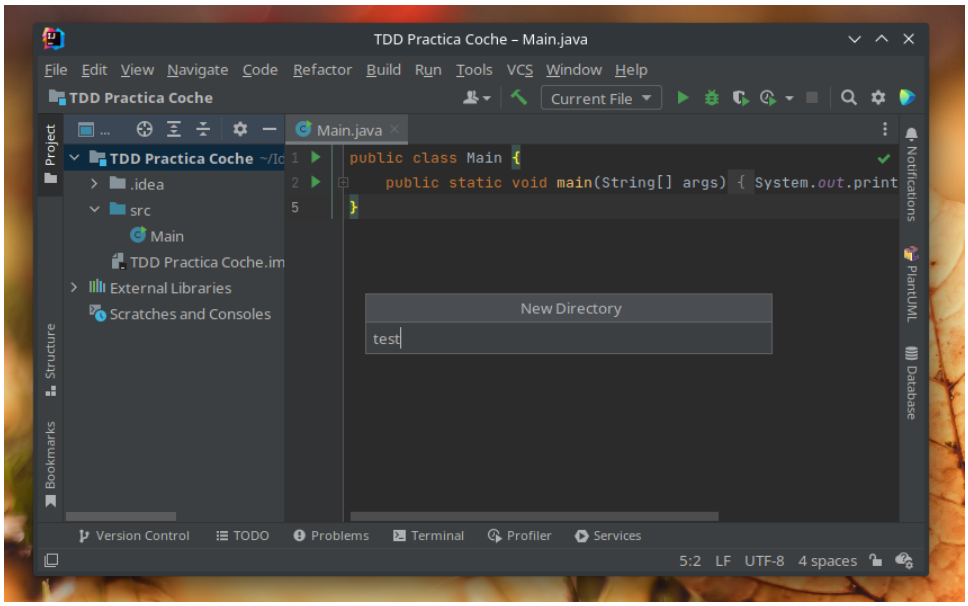
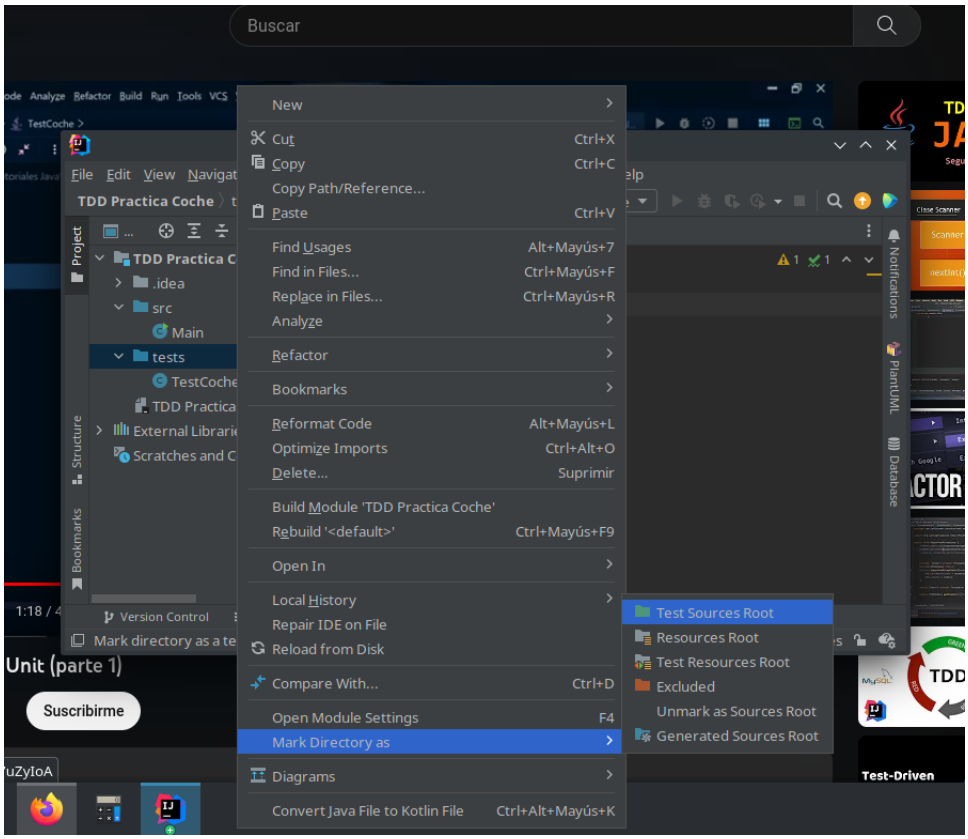
TDD Test Car

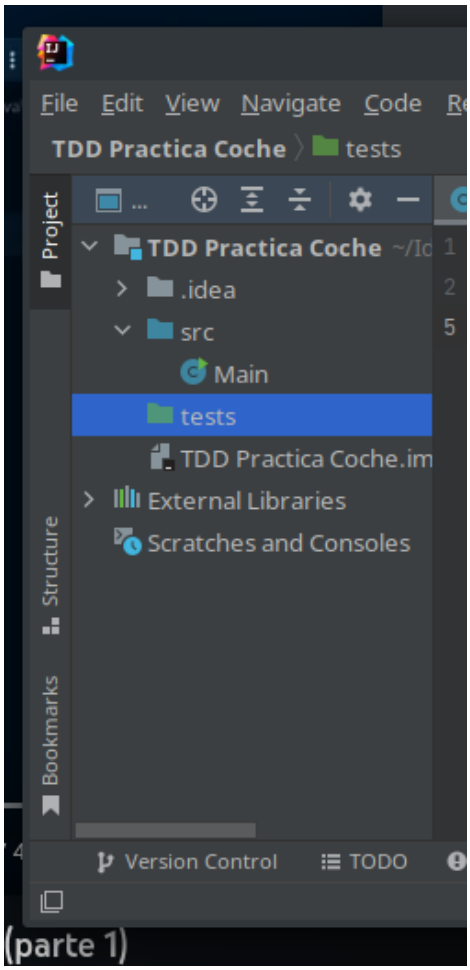
Vamos a crear un nuevo proyecto en IntelliJ para usar la ingienería inversa. Una vez dentro vamos a ir al nombre de nuestro proyecto y vamos a hacer click derecho New/Directory al cual nombraremos tests.



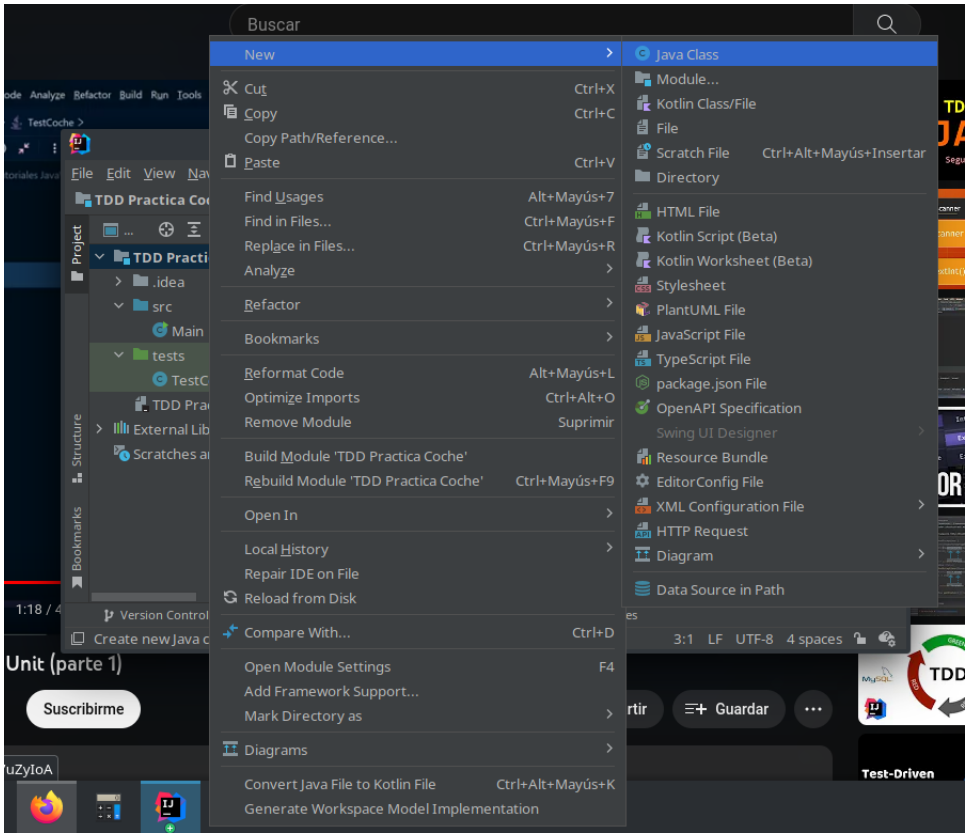
Esta carpeta la vamos a convertir para realizar los test, para ello click derecho sobre el directorio y Mark Directory as/ Test Sources Root.



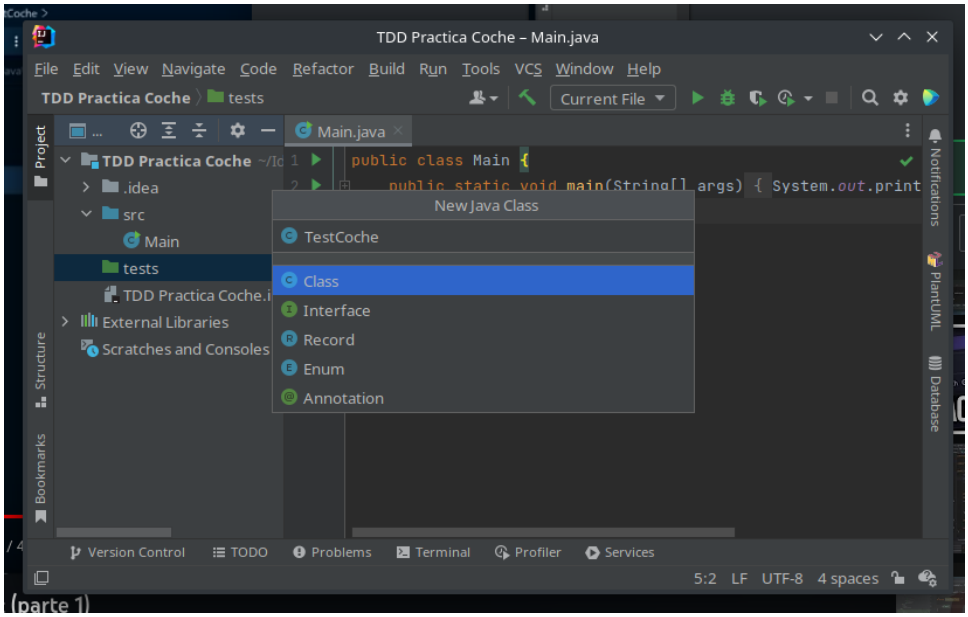
Ahora dicha carpeta nos aparecera en verde indicandonos que es una carpeta de tests.



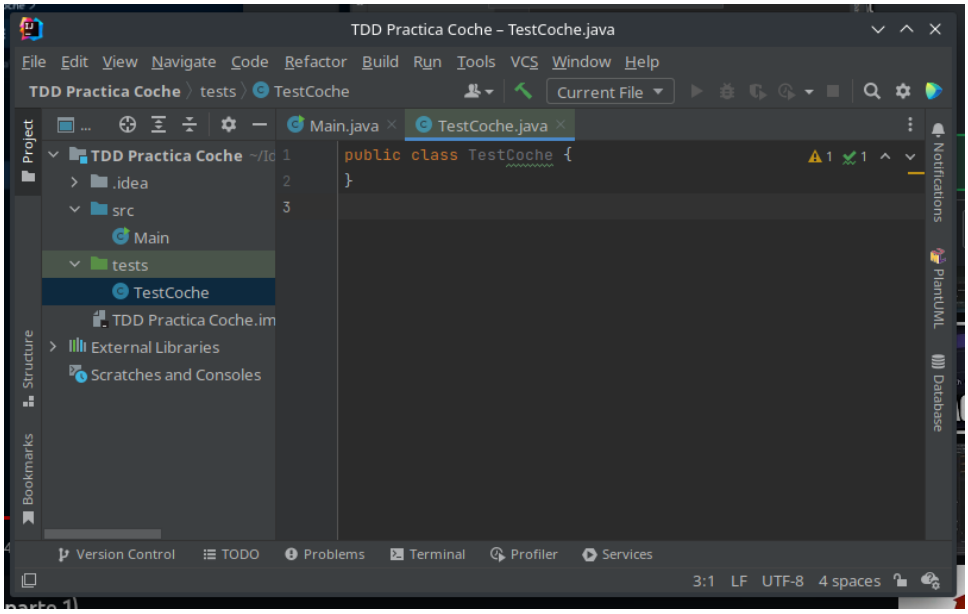
Vamos a crear un nuevo fichero .java en su interior para crear nuestros tests. Para ello click derecho New/Java Class.



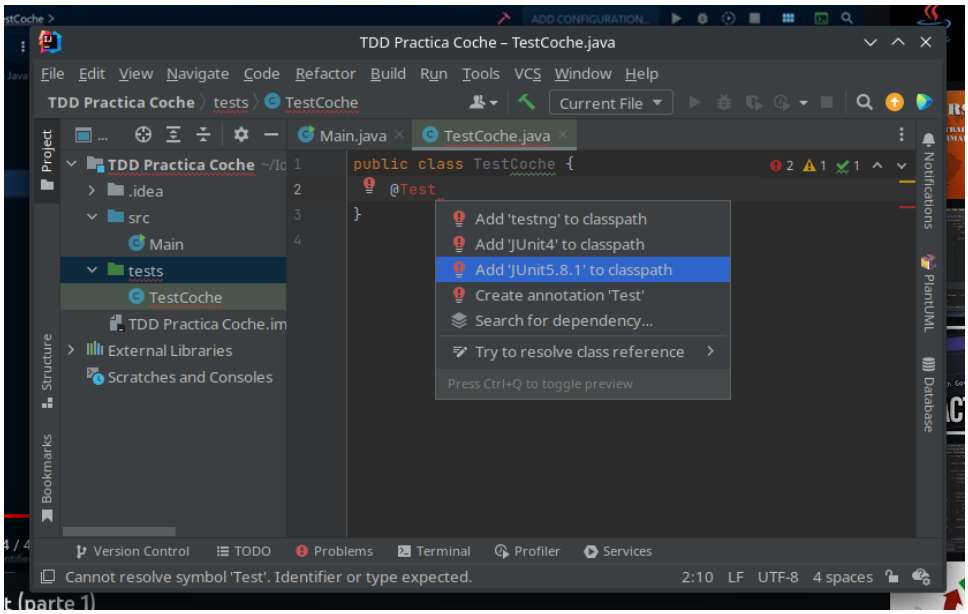
Le pondremos un nombre adecuado para poder identificarlo después.



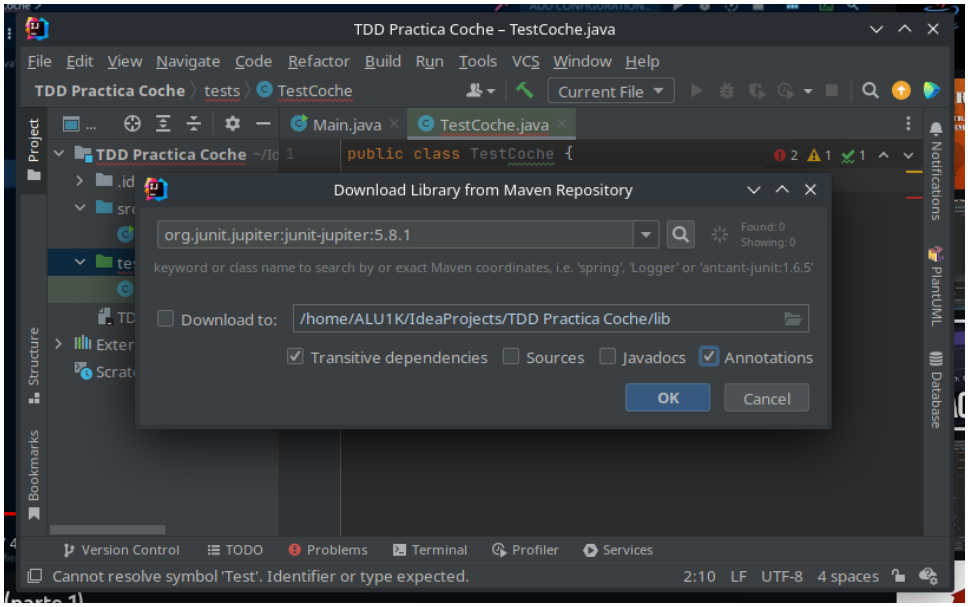
Se nos crea el siguiente archivo .java en el cual vamos a trabajar.



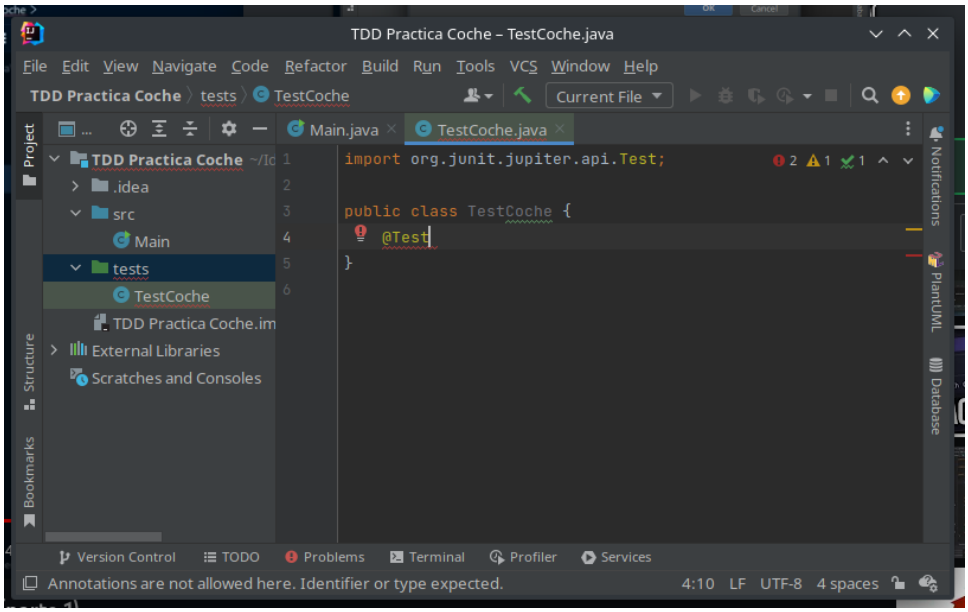
Para ello debemos importar JUnit a nuestro proyecto. Al escribir @Test y hacer alt + enter nos aparecerá el siguiente cuadro de diálogo. Seleccionamos el JUnit deseado.



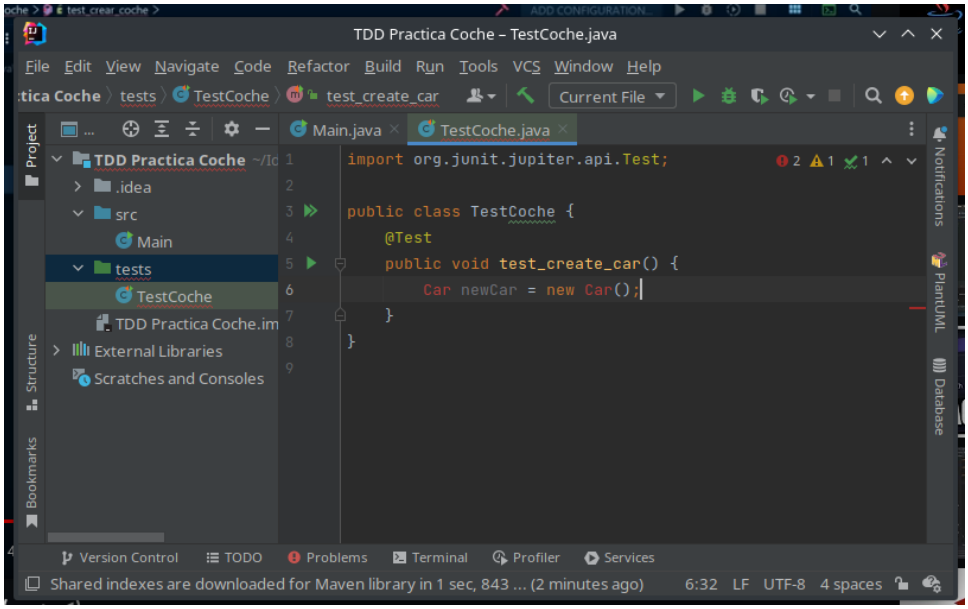
Vamos a seleccionar la pestaña de Annotations antes de aceptar descargar la librería desde el repositorio de Maven.



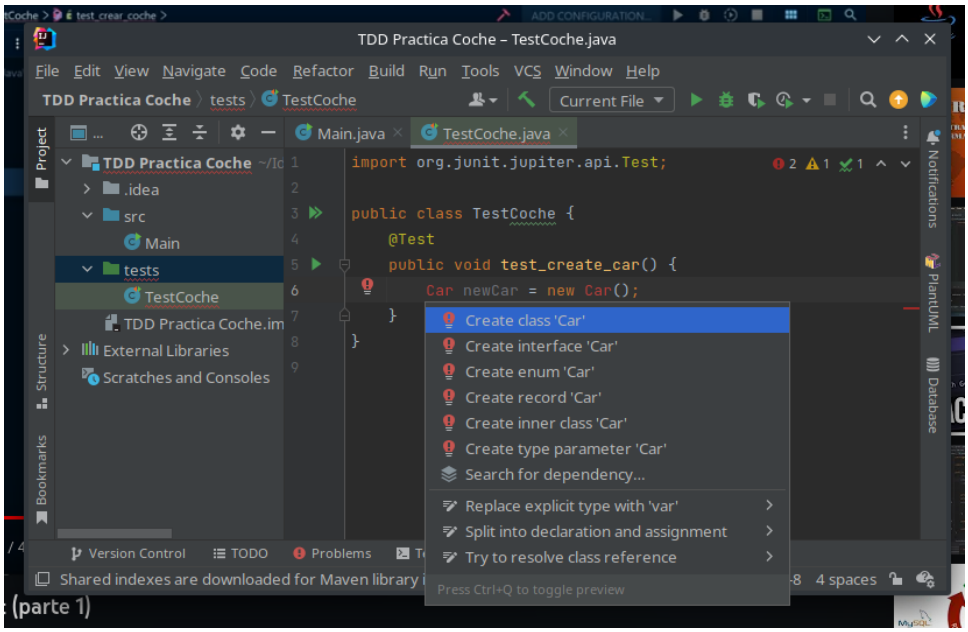
Ahora nuestro @Test nos aparecera en amarillo como que esta implementado.



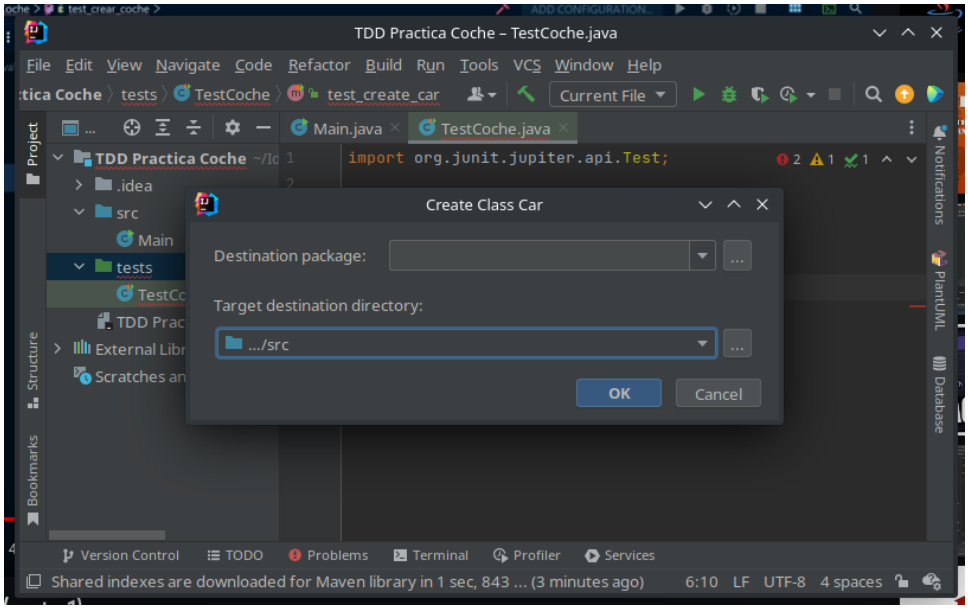
Podemos crear un nuevo método de prueba, vamos a crear uno en el cual crearemos nuevos coches.



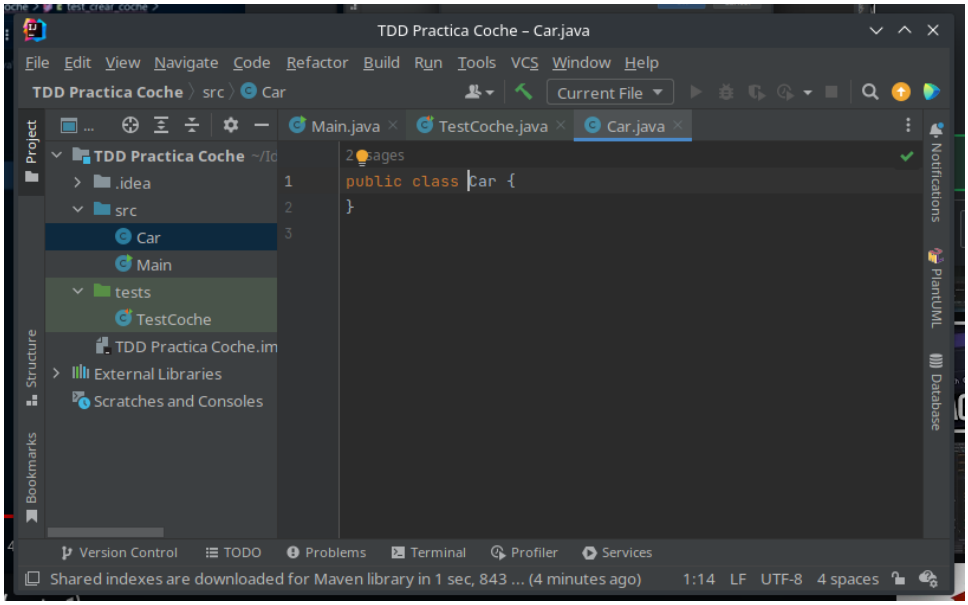
Como estamos desarrollando el proyecto de forma inversa, no tenemos creada la clase Car, por lo que el entorno de desarrollo nos indica que debemos de crearla.



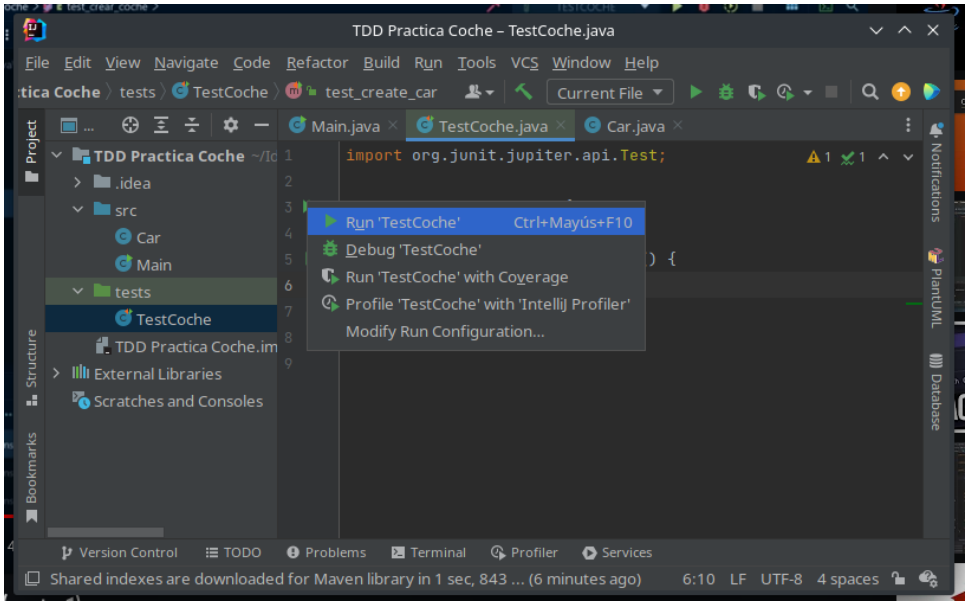
La vamos a localizar en el directorio src que es el que contendrá nuestro proyecto una vez desarrollado



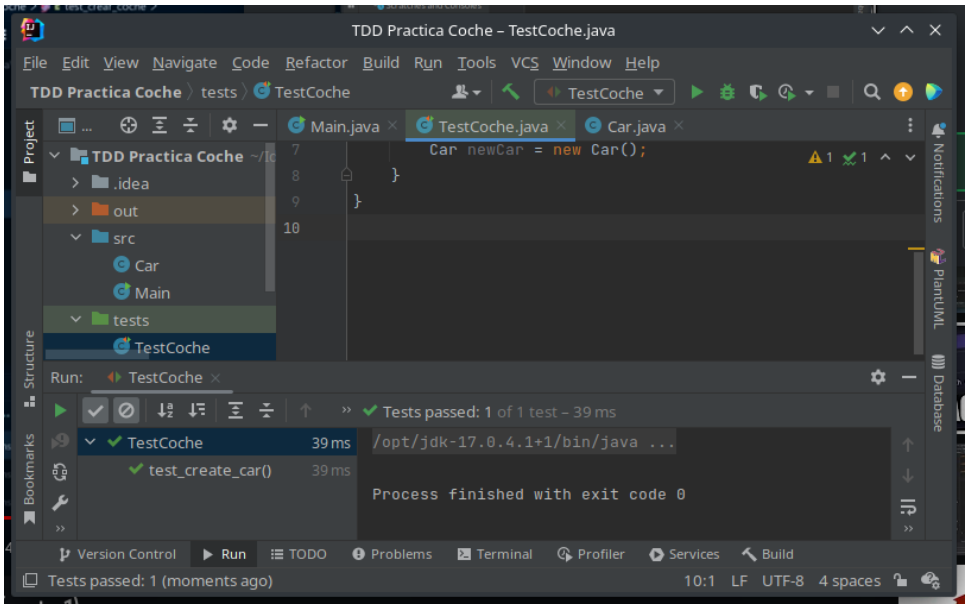
Al crearlo nos aparecerá de la siguiente manera.



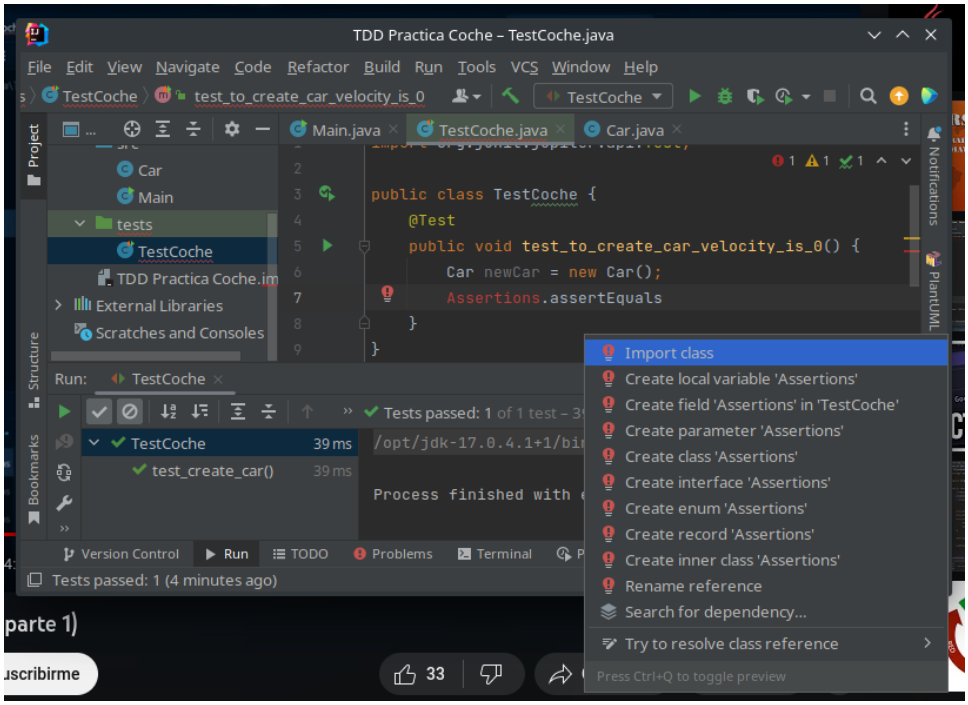
Ahora vamos a hacer una prueba a nuestro test, para eso nos posicionamos en el TestCoche.java vamos a ejecutar el método que queremos probar. Para ello pulsamos el botón de ejecución correspondiente al lado del método.



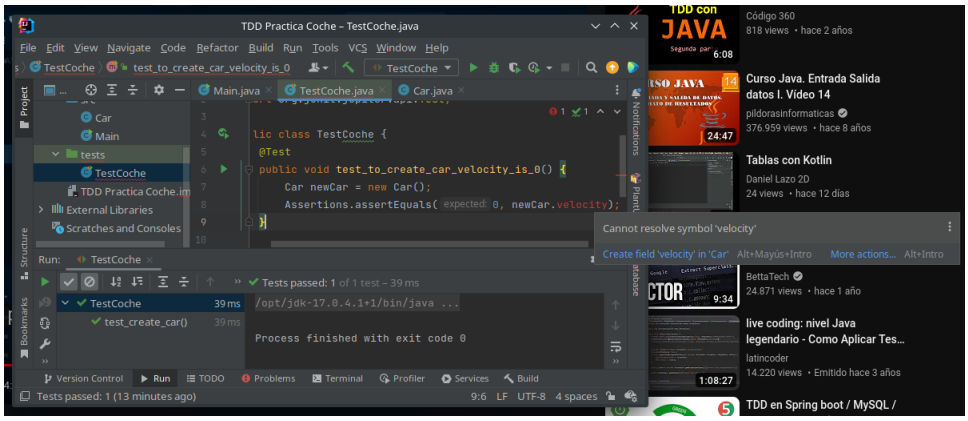
El test ha pasado sin problemas.



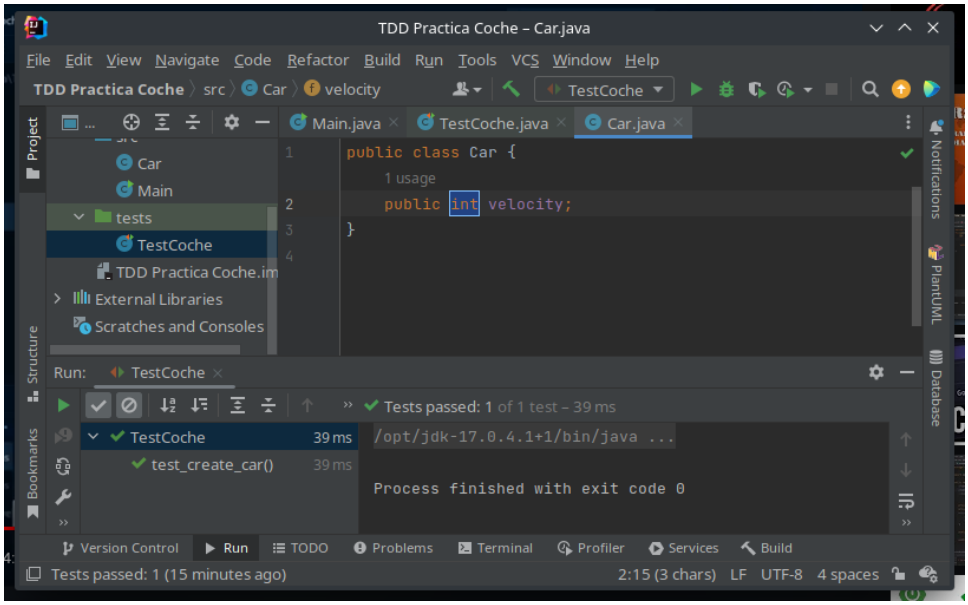
Ahora vamos a modificar el método para que en su creación la velocidad de dicho coche sea 0. Para ello deberemos importar Assertions para hace las comprobaciones y el propio IntelliJ nos lo indicará.



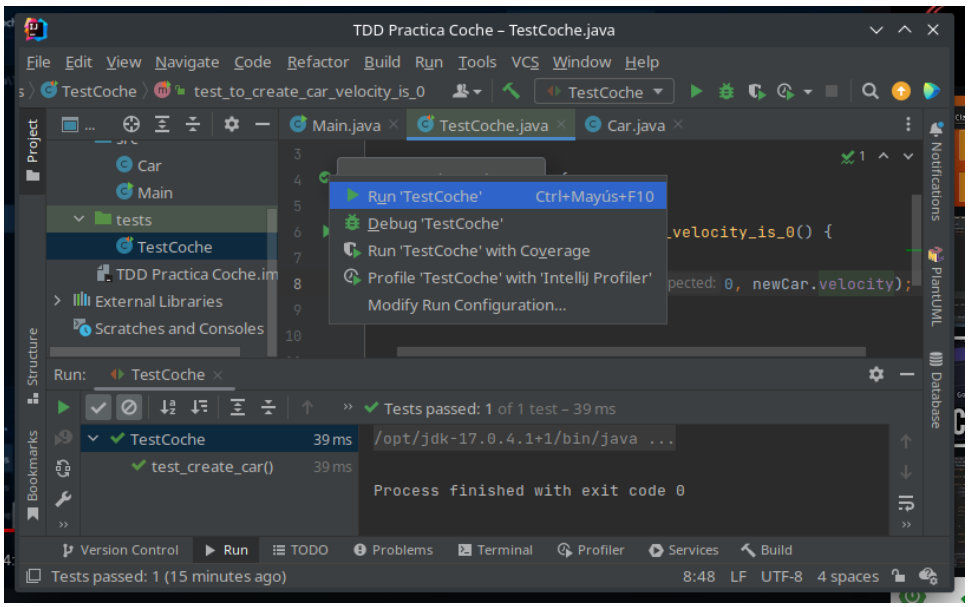
También como vamos a introducir una nueva variable de coche nos indicará que no existe y deberemos de crearla en su clase correspondiente.



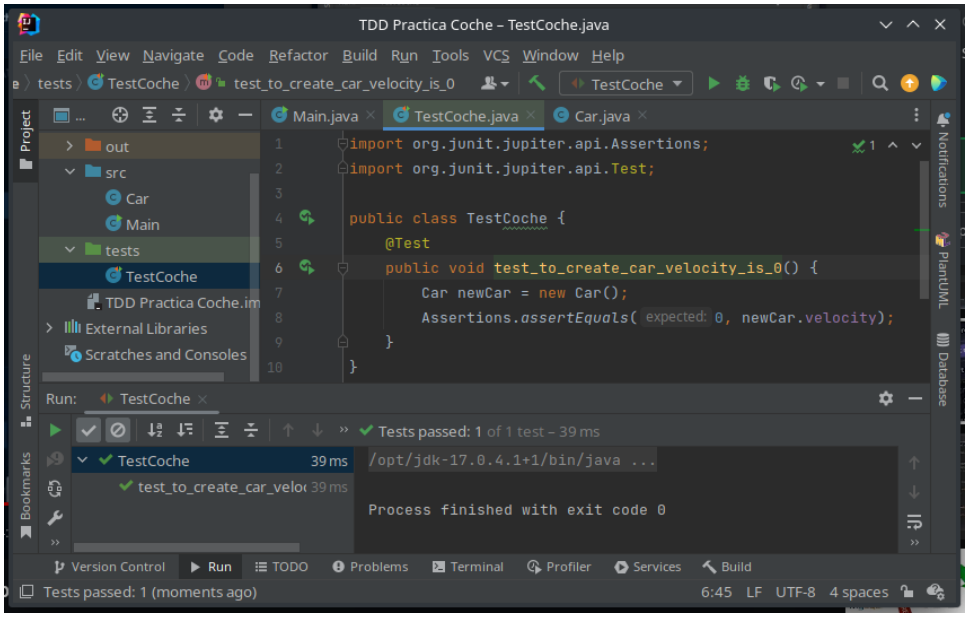
Al crearlo no aparecerá así la variable en su clase, y deberemos de ajustarla a lo que necesitamos.



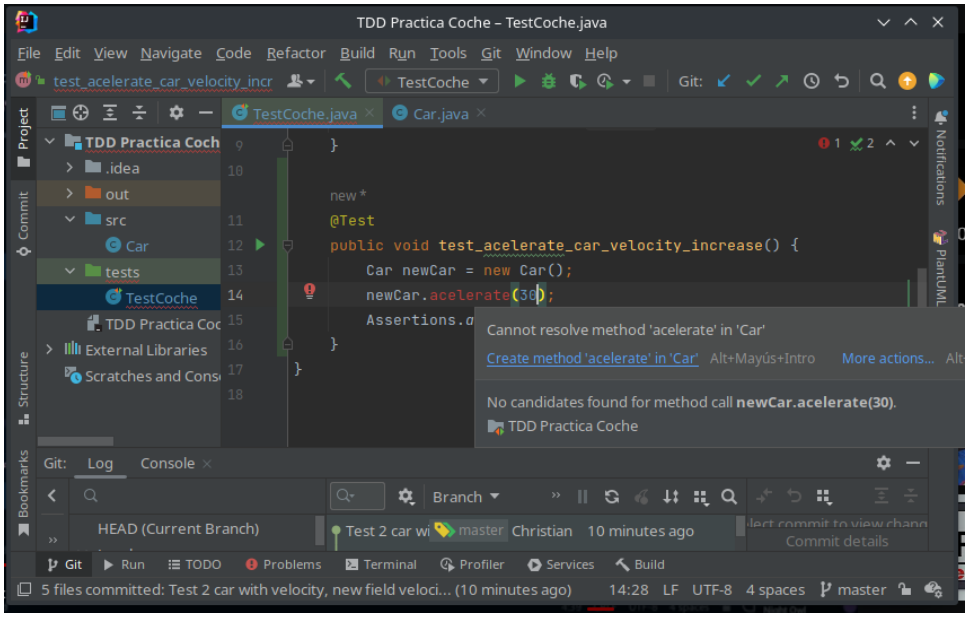
Vamos a ejecutar el método correspondiente en nuestro TestCoche para comprobar que el nuevo método funciona. Seguiremos los pasos anteriores para la comprobación de métodos.



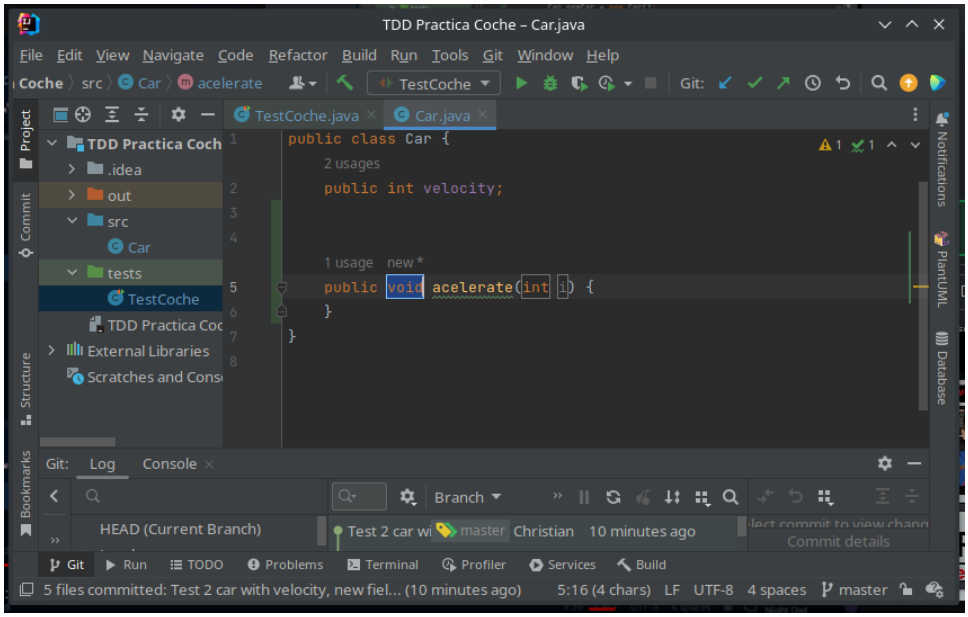
El test ha funcionado correctamentemente de nuevo.



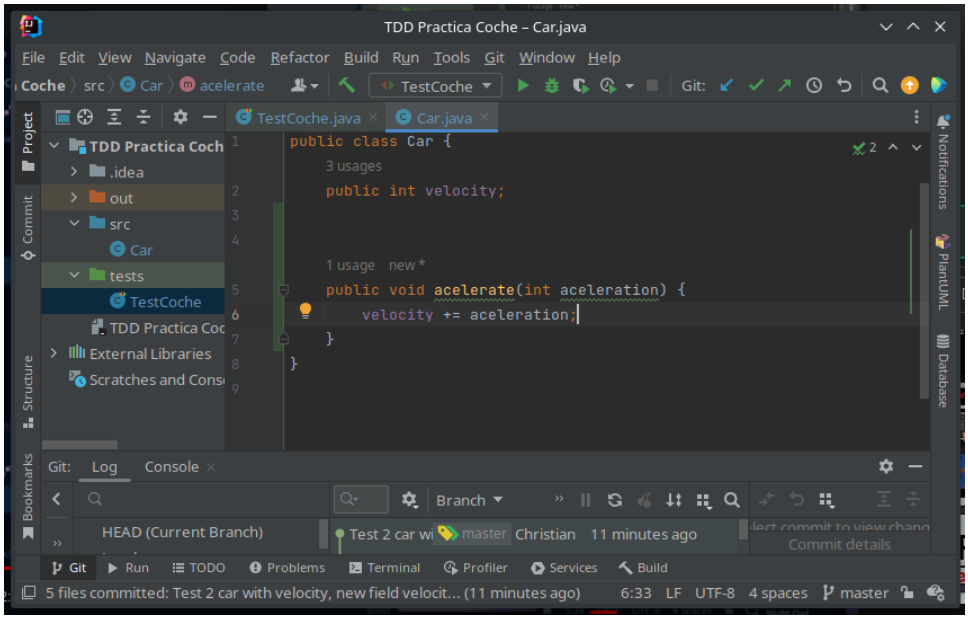
Ahora vamos a crear un método para acelerar nuestro coche. Como no disponemos de dicho método el programa nos lo indicara que debemos crearlo en la clase Car.



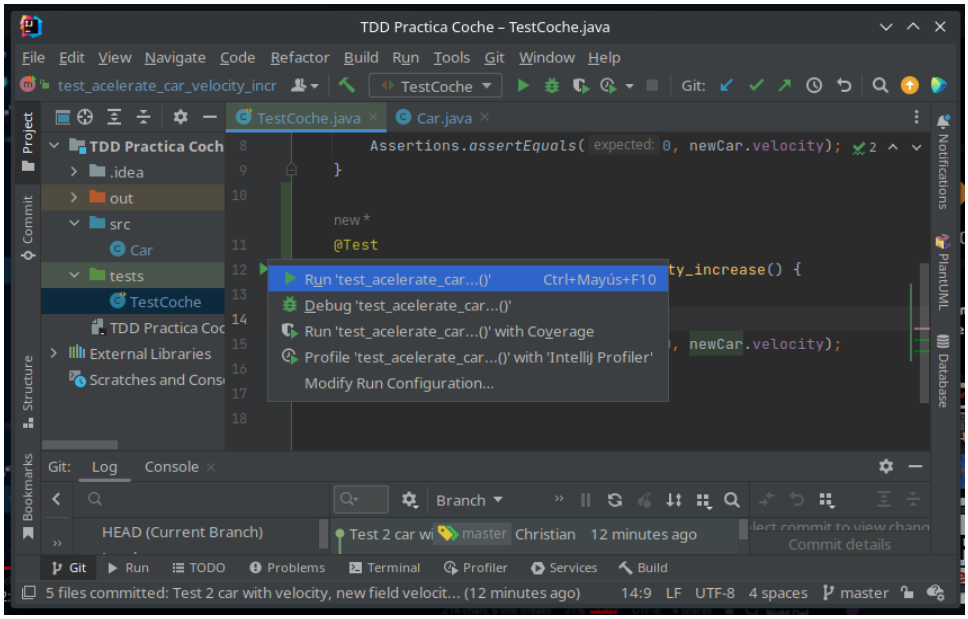
Nos genera el método pero no hace lo que precisamos.



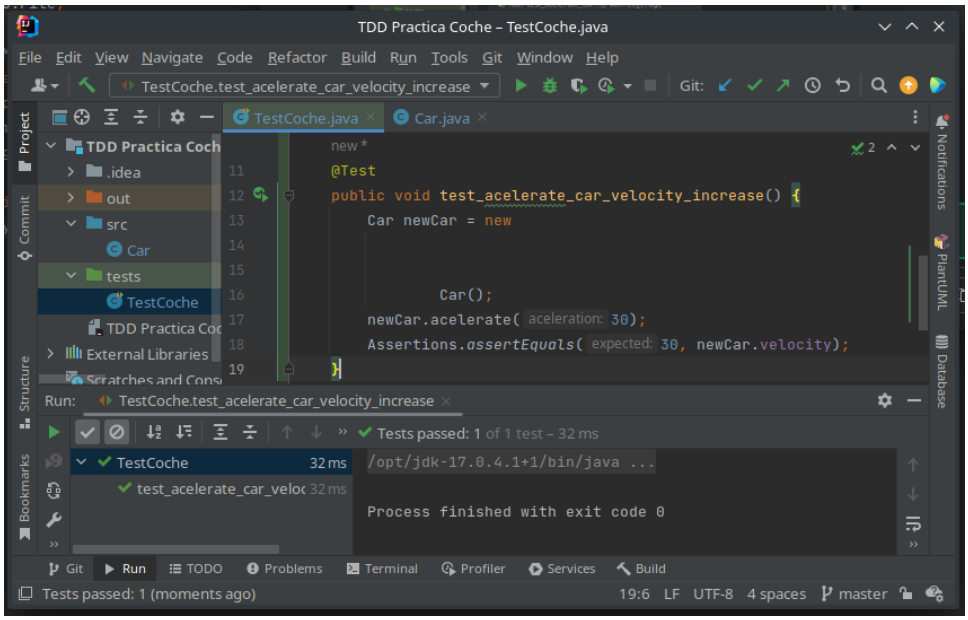
Debemos modificarlo de la siguiente manera para que sea acorde con lo que tenemos que realizar.



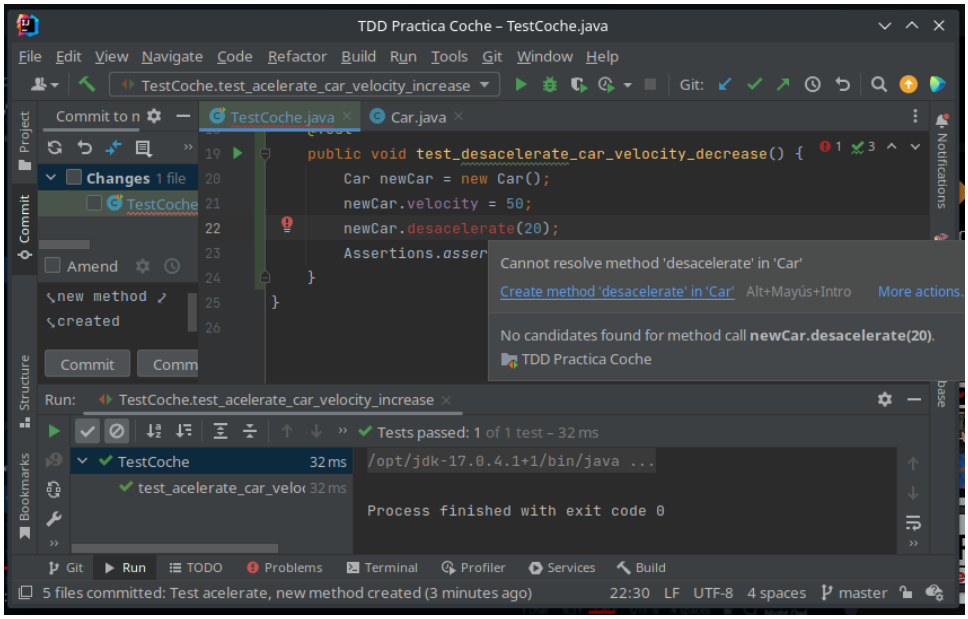
Una vez realizado los cambios necesarios lo comprobaremos.



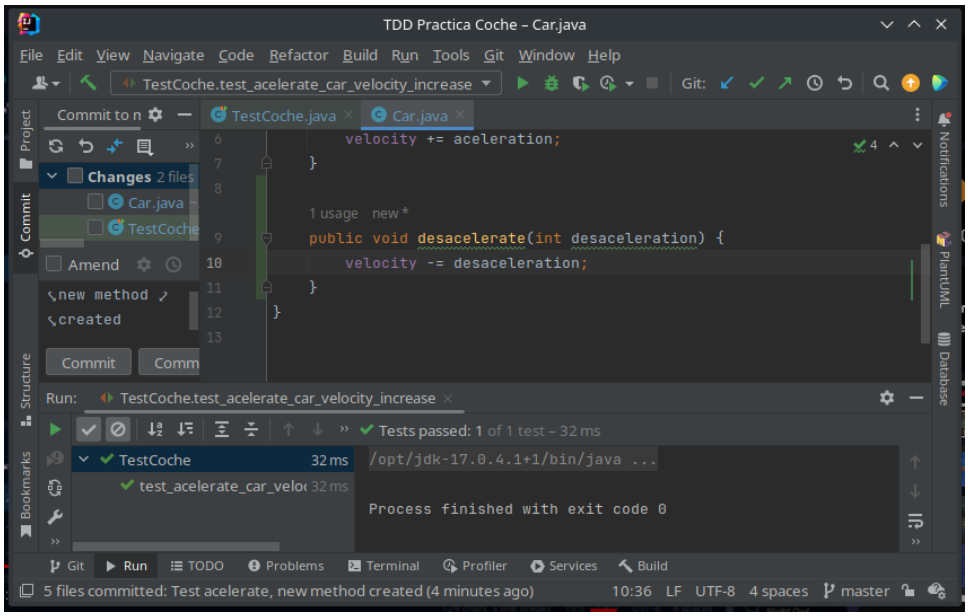
Ha pasado el test correctamente.



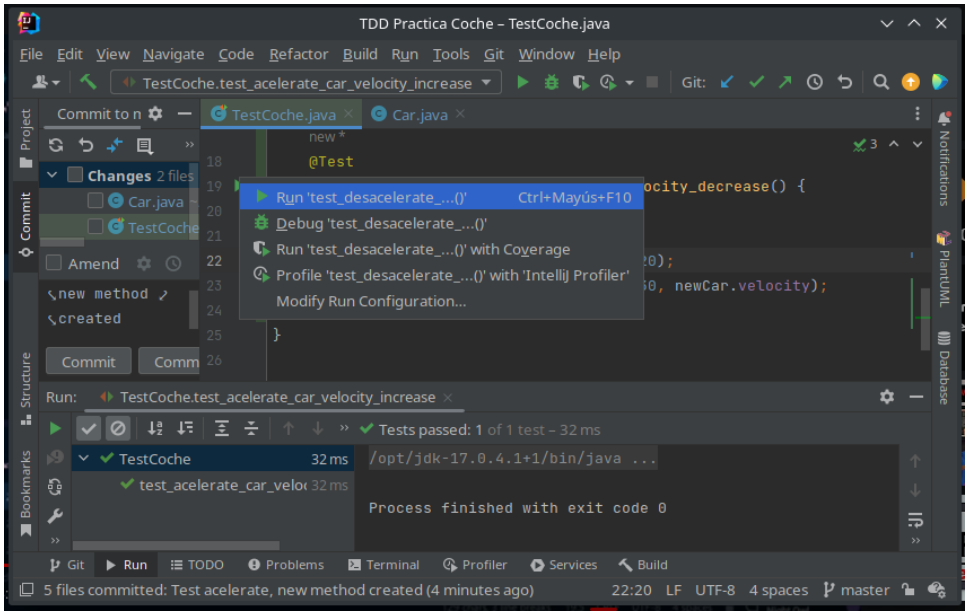
Ahora vamos a realizar otro método para desacelerar nuestro coche, como no tenemos el método correspondiente vamos a tener que generarlo.



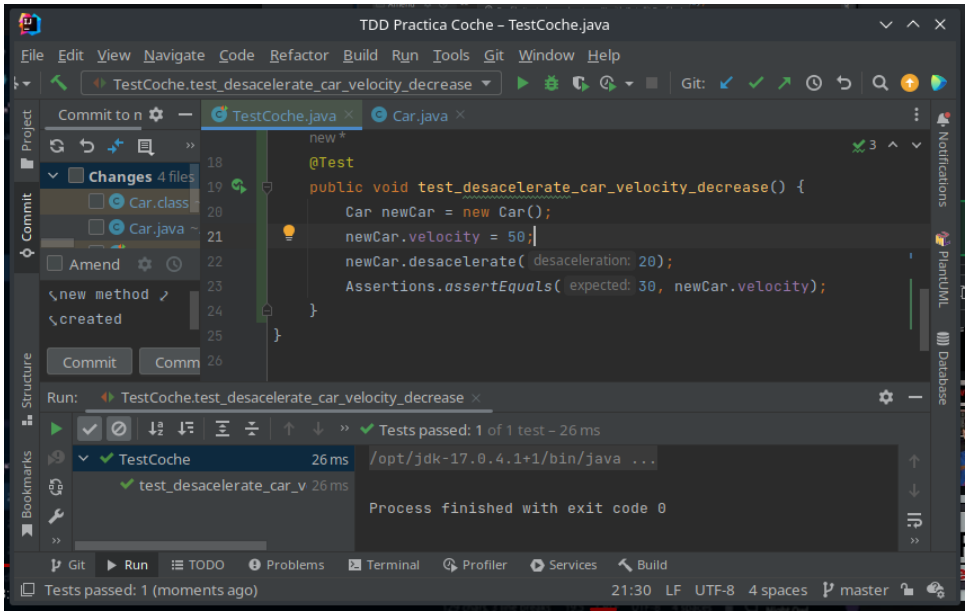
Tendremos que volver a modificarlo para que haga lo que necesitamos.



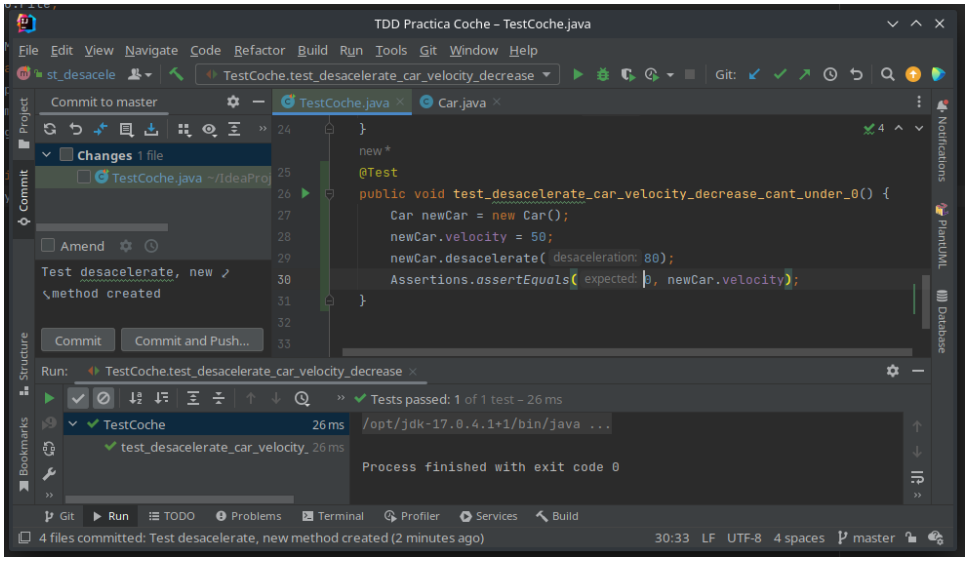
Y lo mandaremos a realizar la prueba, poniendo en el assert los valores correspondientes.



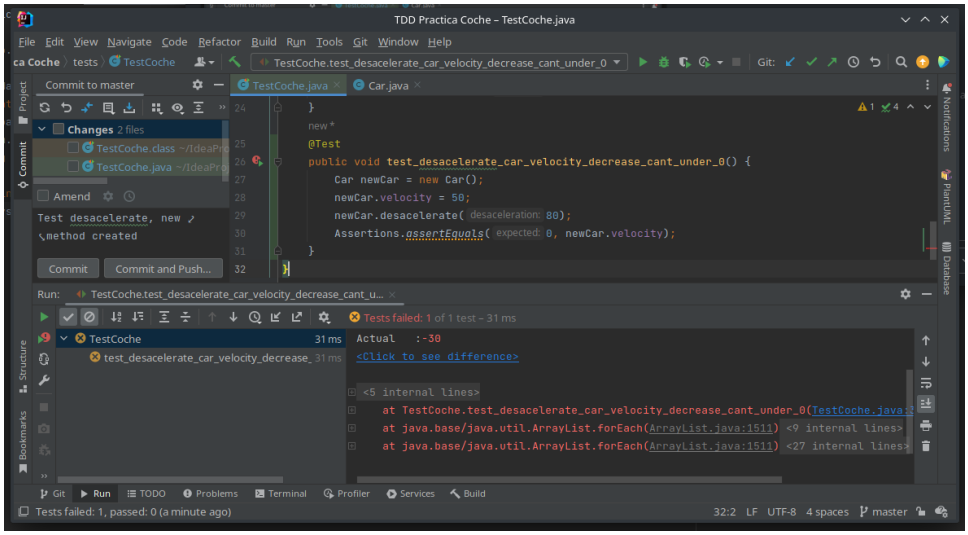
Como los parámetros que hemos introducid son correctos el test se llevará a cabo correctamente.



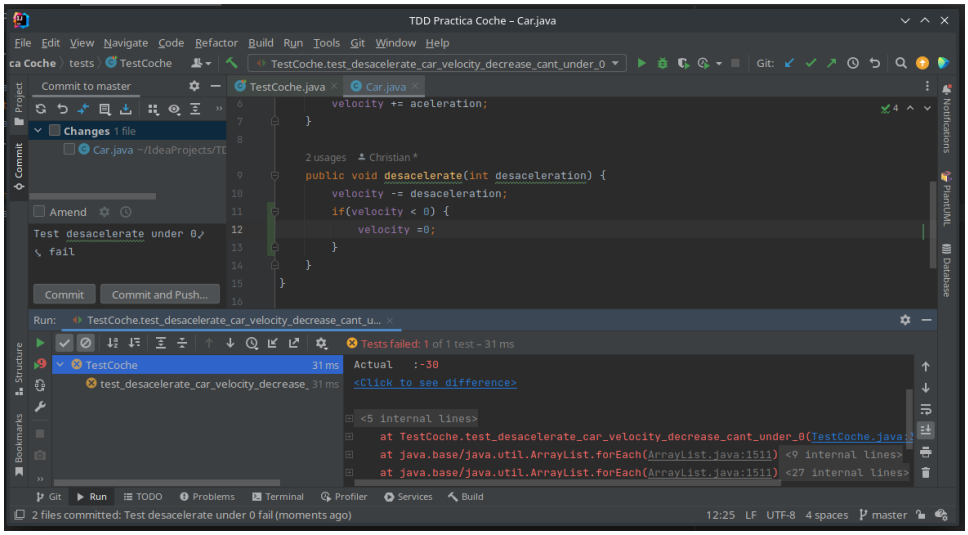
Vamos a modificar ahora el resultado esperado, ya que un coche no deberia tener una velocidad negativa.



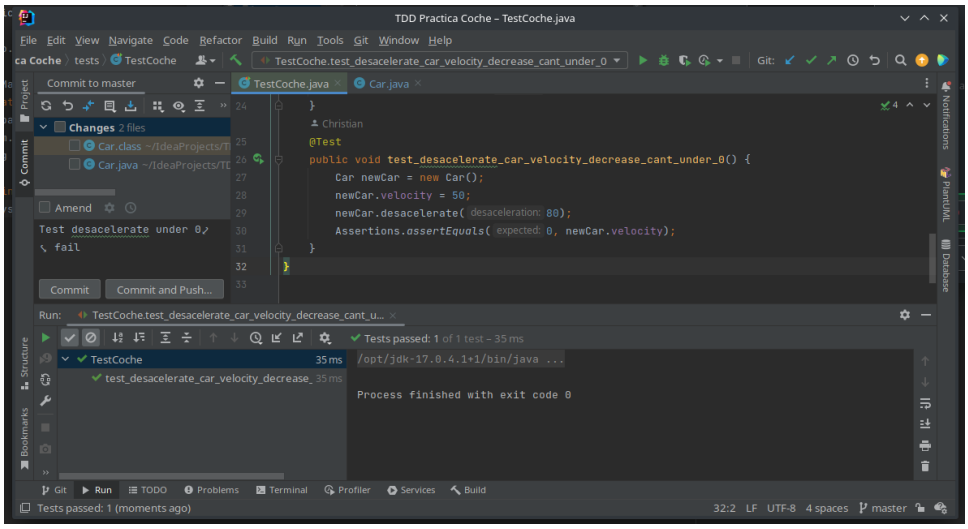
Podremos compilar el método sin problemas, pero como el resultado esperado no corresponde con el que nos devuelve el método nos aparece una discrepancia y nos lo indica el programa.



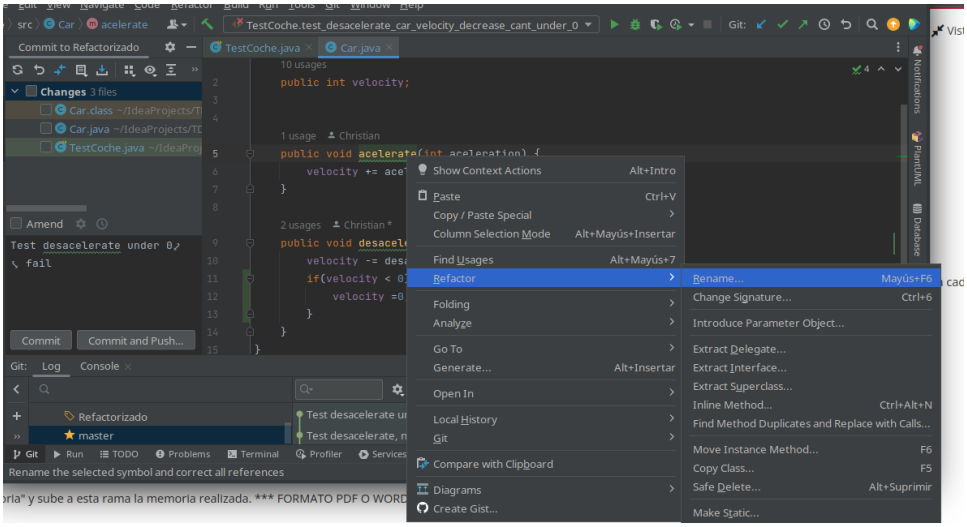
Para solucionar este problema no contemplado por el entorno vamos a tener que añadir un condicionar a nuestro método, indicando que si la velocidad se reduce por debajo de 0 no será posible y detendrá nuestro coche dejando la velocidad en 0.



Ahora al pasar nuestro test de nuevo comprobamos que funciona correctamente.



Esto es la segunda parte del ejercicio que pide que una rama aparte realicemos Refactor del nombre de los métodos para indicar que lo hemos realizado nosotros.


Commit to Refactorizado
Changes 3 files
Car.class ~/IdeaProjects/T
Car.java ~/IdeaProjects/TE
TestCoche.java ~/IdeaProj
Amend
Test desacelerate under 0
fail
Commit
Commit and Push...
TestCoche.java
Car.java
10 usages
public int velocity;
1 usage Christian
public void acelerate(int aceleration) {
velocity += ace
}
2 usages Christian *
public void desacele
velocity -= des
if(velocity < 0
velocity =0
}
}
Git: Log Console
Refactorizado
master
Test desacelerate ur
Test desacelerate, n
Git Run TODO Problems Terminal Profiler Services
Rename the selected symbol and correct all references
Show Context Actions Alt+Intro
Paste Ctrl+V
Copy / Paste Special
Column Selection Mode Alt+Mayús+Insertar
Find Usages Alt+Mayús+7
Refactor
Folding
Analyze
Go To
Generate... Alt+Insertar
Open In
Local History
Git
Compare with Clipboard
Diagrams
Create Gist...
Rename... Mayús+F6
Change Signature... Ctrl+6
Introduce Parameter Object...
Extract Delegate...
Extract Interface...
Extract Superclass...
Inline Method... Ctrl+Alt+N
Find Method Duplicates and Replace with Calls...
Move Instance Method... F6
Copy Class... F5
Safe Delete... Alt+Suprimir
Make Static...
Notifications
PlantUML
Database