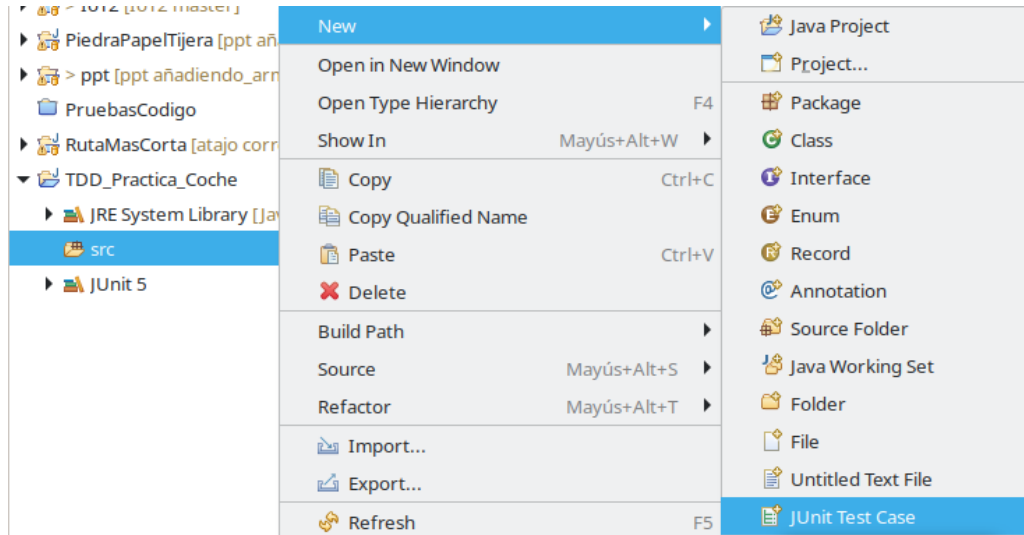
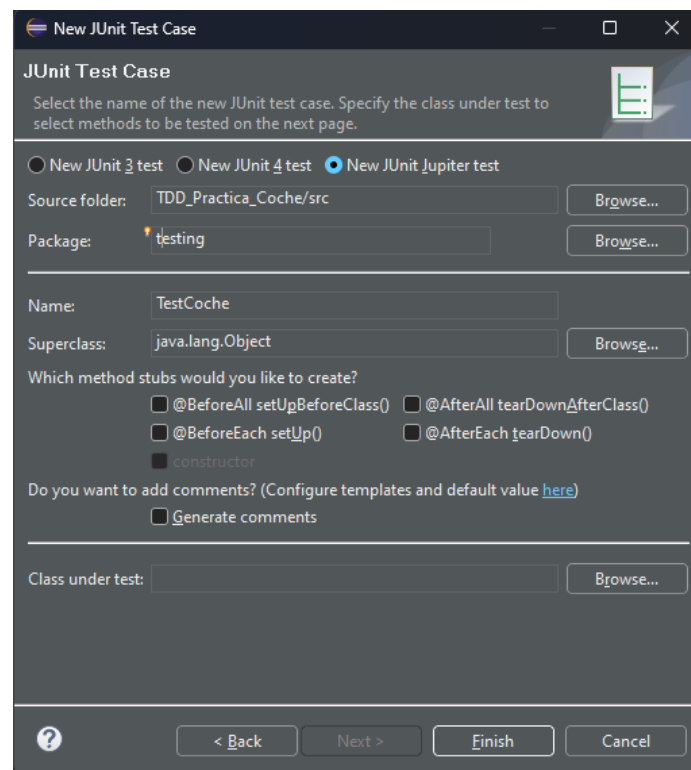


# Creando TDD en Eclipse

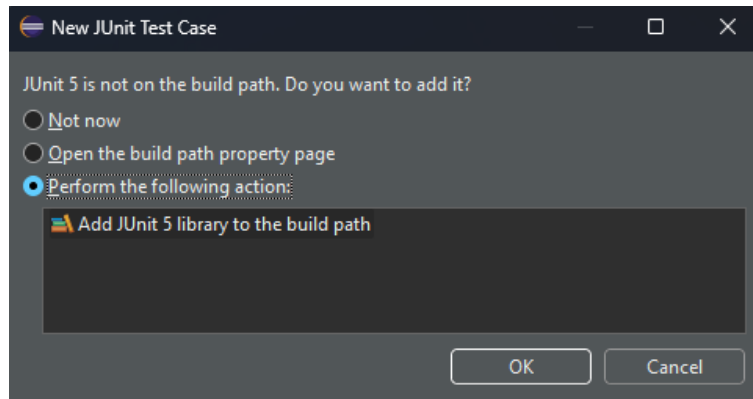
El primer paso es crear un nuevo proyecto Java. Después haremos click derecho sobre la carpeta “src” y seleccionaremos “New →JUnit Test Case”.



Se nos abrirá una ventana para configurar el “JUnit Test Case”, la rellenaremos con lo que nos aparece en esta imagen:



Al hacer click en “Finish” nos preguntará si queremos añadir la las librerías de JUnit 5, le daremos a “Perform de following action” y a “OK”.



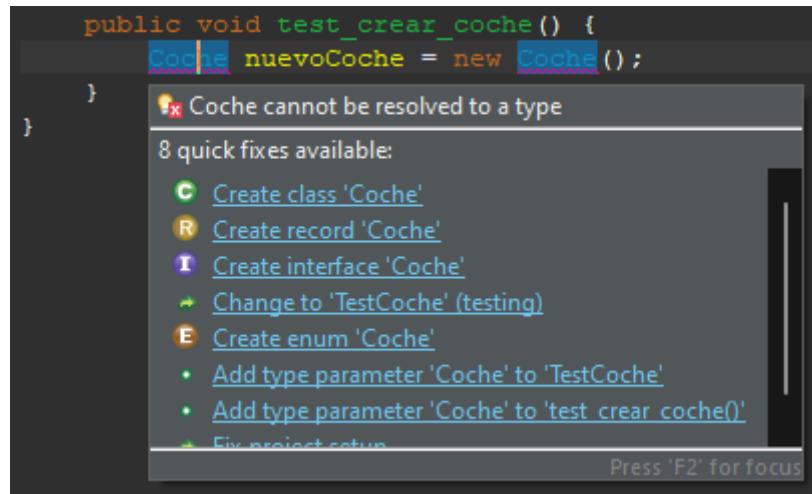
Una vez creado el archivo, nos habrá generado este código de forma automática:

```
TestCoche.java X
1 package testing;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class TestCoche {
8
9     @Test
10    void test() {
11        fail("Not yet implemented");
12    }
13
14 }
15
```

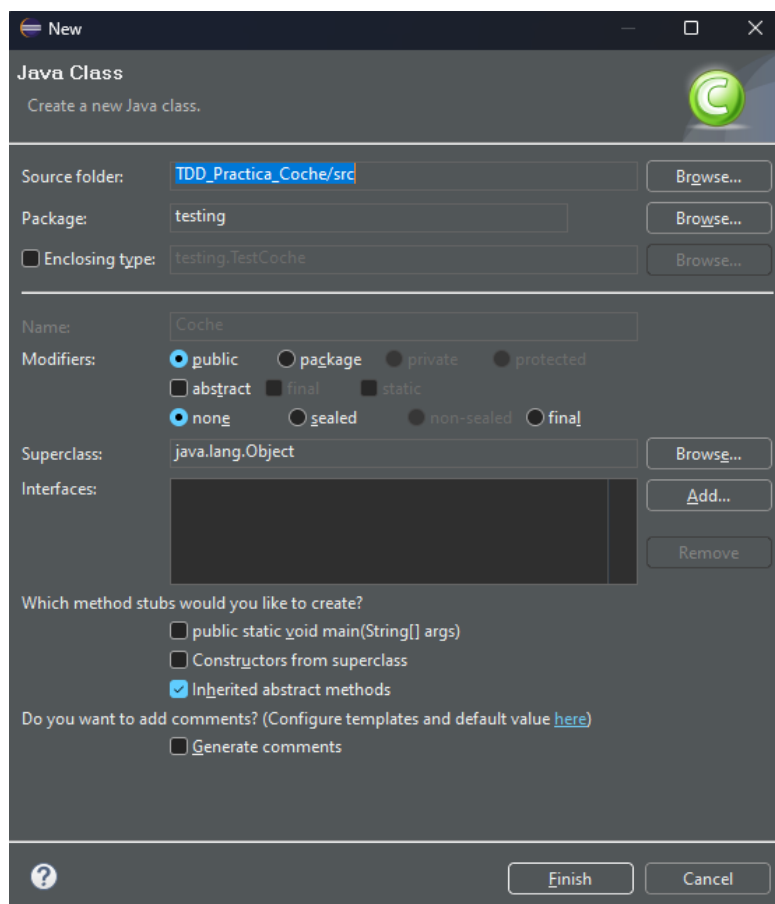
Ahora vamos a modificar el código para empezar a hacer tests. Quitaremos el método que nos ha creado por defecto y pondremos este:

```
TestCoche.java X
1 package testing;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6
7 class TestCoche {
8
9     @Test
10    public void test_crear_coche() {
11        Coche nuevoCoche = new Coche();
12    }
13 }
```

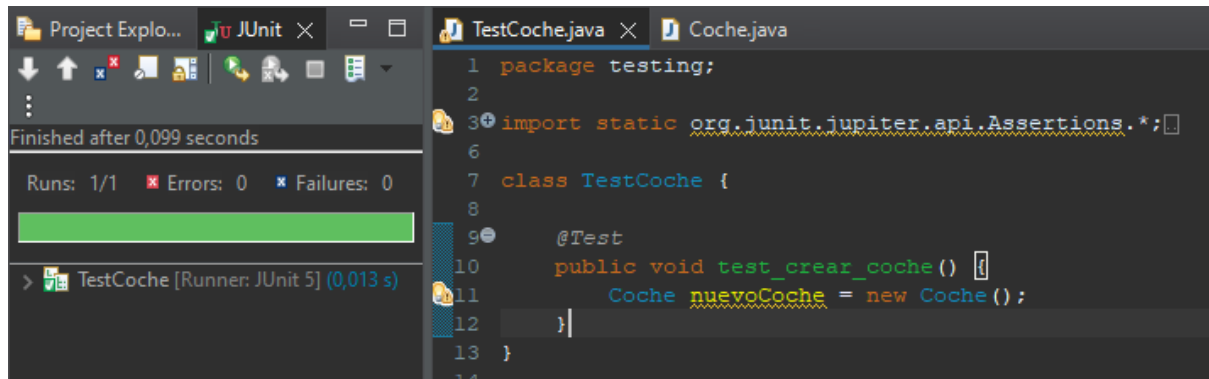
Como vemos, hemos intentado crear un objeto de la clase “Coche”, pero nos salta un error porque no hemos creado ninguna clase “Coche” todavía. Vamos a hacer que nos la cree Eclipse, si dejamos el ratón quieto sobre la palabra Coche nos aparecerán las siguientes opciones:



Seleccionaremos la primera opción “Create class ‘Coche’” y nos aparecerá la siguiente ventana. La rellenaremos como aparece en esta imagen:



Una vez hecho esto podemos probar a ejecutar el test y veremos que nos pasa el test sin problemas.



Ahora vamos a complicarlo un poco más ya que este ejemplo es demasiado simple. Cambiaremos el nombre al método:

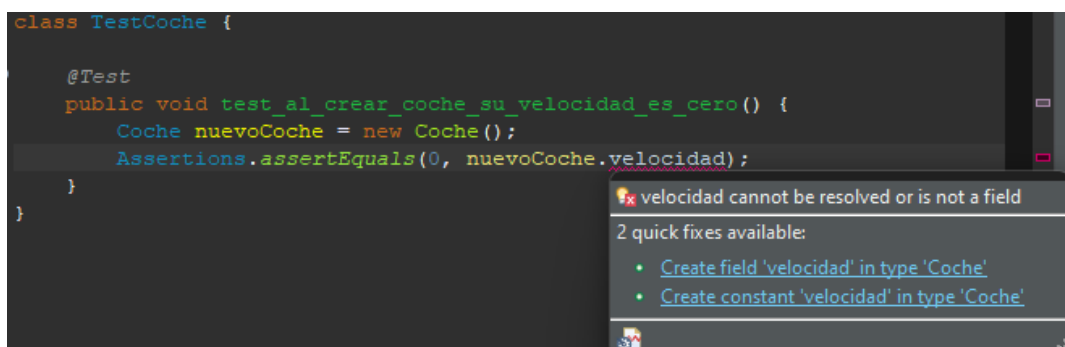
“test\_crear\_coche” por “test\_al\_crear\_coche\_su\_velocidad\_es\_cero”.

```
class TestCoche {

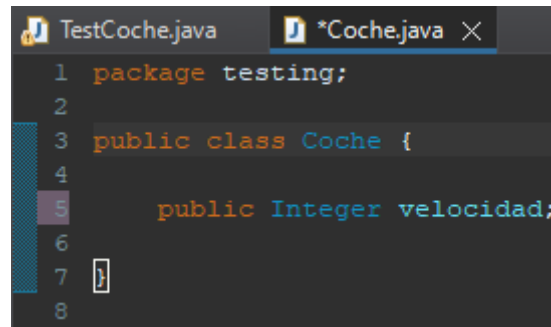
    @Test
    public void test_al_crear_coche_su_velocidad_es_cero() {
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }

}
```

Como vemos en la imagen anterior, he añadido de forma manual una nueva línea de código. La función de esta nueva línea es comprobar que la velocidad es cero nada más crear el objeto “nuevoCoche”. Pero para ello primero debemos crear un atributo nuevo llamado “velocidad” y decir que valga 0, todo esto en la clase que hemos creado antes automáticamente. Lo podemos hacer de forma automática de la misma forma que lo hemos hecho antes, dejando el ratón sobre la palabra “velocidad”.

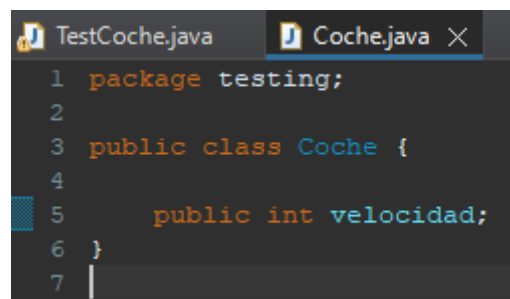


Seleccionamos la opción “Create field ‘velocidad’ in type ‘Coche’” y nos habrá generado este código en la clase “Coche”.



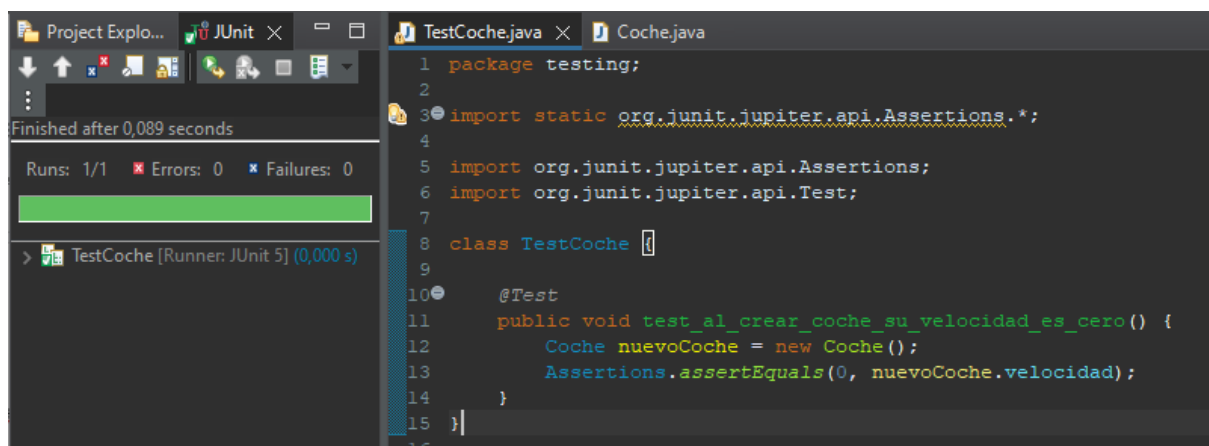
```
1 package testing;
2
3 public class Coche {
4
5     public Integer velocidad;
6
7 }
8
```

Nos ha creado automáticamente un atributo velocidad de tipo “Integer”, nosotros no necesitamos que sea de tipo “Integer” así que lo cambiaremos a “int”.



```
1 package testing;
2
3 public class Coche {
4
5     public int velocidad;
6 }
7
```

Ahora ejecutamos, y si hemos seguido los pasos correctamente deberíamos pasar el test sin ningún problema.



```
1 package testing;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 import org.junit.jupiter.api.Assertions;
6 import org.junit.jupiter.api.Test;
7
8 class TestCoche {
9
10     @Test
11     public void test_al_crear_coche_su_velocidad_es_cero() {
12         Coche nuevoCoche = new Coche();
13         Assertions.assertEquals(0, nuevoCoche.velocidad);
14     }
15 }
16
```

Project Explo... JUnit X

Finished after 0,089 seconds

Runs: 1/1 Errors: 0 Failures: 0

> TestCoche [Runner: JUnit 5] (0,000 s)

Ahora crearemos un nuevo método llamado:

**“test\_al\_acelerar\_un\_coche\_su\_velocidad\_aumenta”.**

En este método lo que haremos será testear que la velocidad del coche aumenta cuando utilicemos el método acelerar que vamos a crear en la clase Coche.

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

Como vemos nos salta un error ya que en la clase “Coche” no tenemos ningún método que se llame acelerar, vamos a crearlo de manera automática igual que hemos creado el atributo velocidad.

```
public void test_al_acelerar_un_coche_su_velocidad_aumenta() {
    Coche nuevoCoche = new Coche();
    nuevoCoche.acelerar(30);
    Assertions.assertEquals(30, nuevoCoche.velocidad);
}
```

The method `acelerar(int)` is undefined for the type `Coche`

2 quick fixes available:

- Create method '`acelerar(int)`' in type '`Coche`'
- Add cast to '`nuevoCoche`'

Press 'F2' for focus

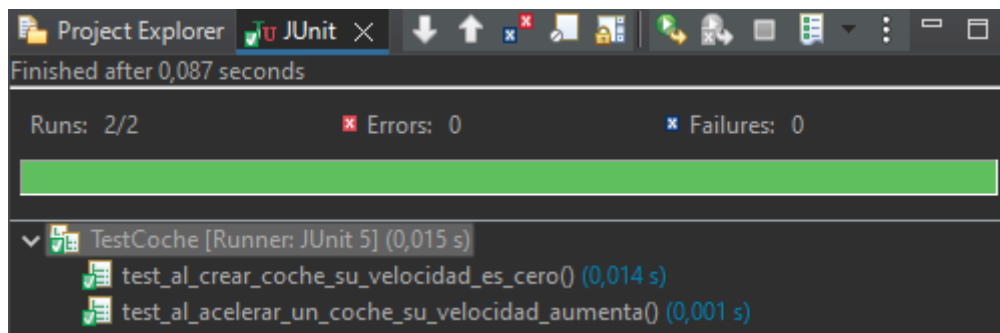
Pinchamos en “Create method ‘acelerar(int)’ in type ‘Coche’” y se nos generará lo siguiente en la clase “Coche”.

```
TestCoche.java  *Coche.java X
1 package testing;
2
3 public class Coche {
4
5     public int velocidad;
6
7     public void acelerar(int i) {}
8     // TODO Auto-generated method stub
9
10 }
```

Modificaremos esto para que al llamar a este método la velocidad del coche aumente.

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

Ahora, si ejecutamos el test, deberíamos pasarlo correctamente.



El siguiente paso será crear un nuevo método para comprobar la deceleración del coche, este método se llamará:

**“test\_al\_decelerar\_un\_coche\_su\_velocidad\_disminuye”.**

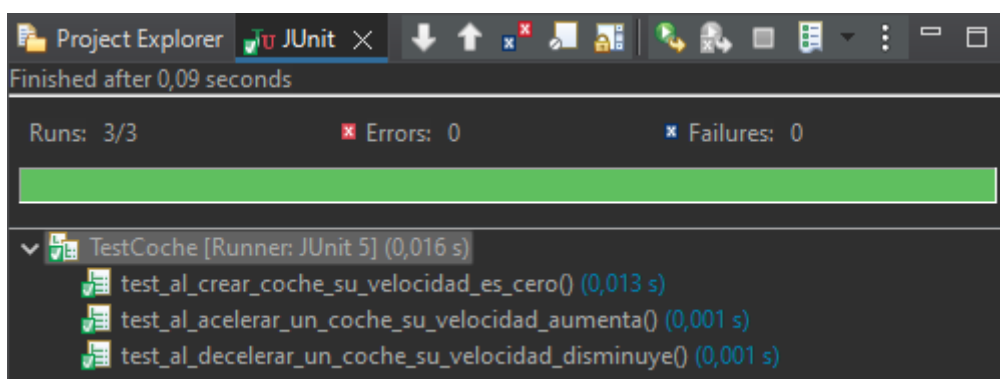
Este método comprobará que el método que vamos a crear en la clase Coche llamado “decelerar” funciona correctamente.

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

Para crear el método “decelerar” lo podemos crear de la misma forma que hemos creado el método acelerar. Finalmente quedaría así después de realizarle las modificaciones necesarias para este test.

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

Ahora, si ejecutamos el test, deberíamos pasarlo correctamente.



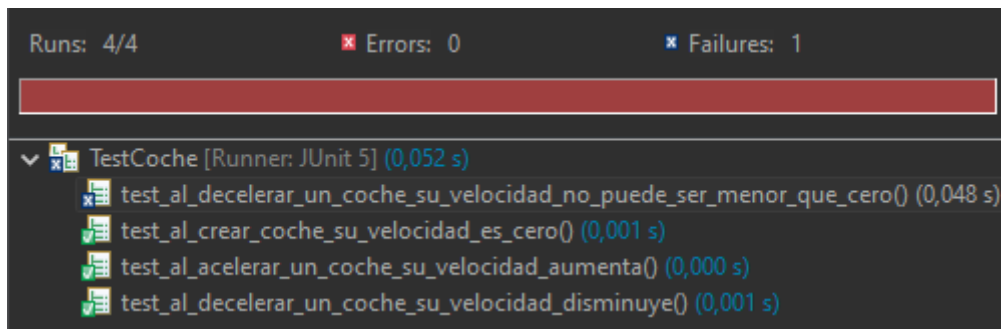
Ahora crearemos otro nuevo método de test que se llamará:

**“test\_al\_decelerar\_un\_coche\_su\_velocidad\_no\_puede\_ser\_menor\_que\_cero”**

Con este método comprobaremos que la velocidad no puede ser menor que cero al decelerar un coche. El código de este método de test sería el siguiente:

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

Si ejecutamos ahora vemos que no habremos pasado el test en este método. Esto se debe a que el Assertions estaba esperando un 0 pero recibió un 30.



Esto se debe a que no hemos modificado el código del método “decelerar” en la clase “Coche” para que su velocidad no pueda ser menor que 0. Para solucionar esto hay que modificar el código del método “decelerar” creado anteriormente.

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

Con esta modificación realizada ejecutamos de nuevo y ya nos funcionará correctamente.

