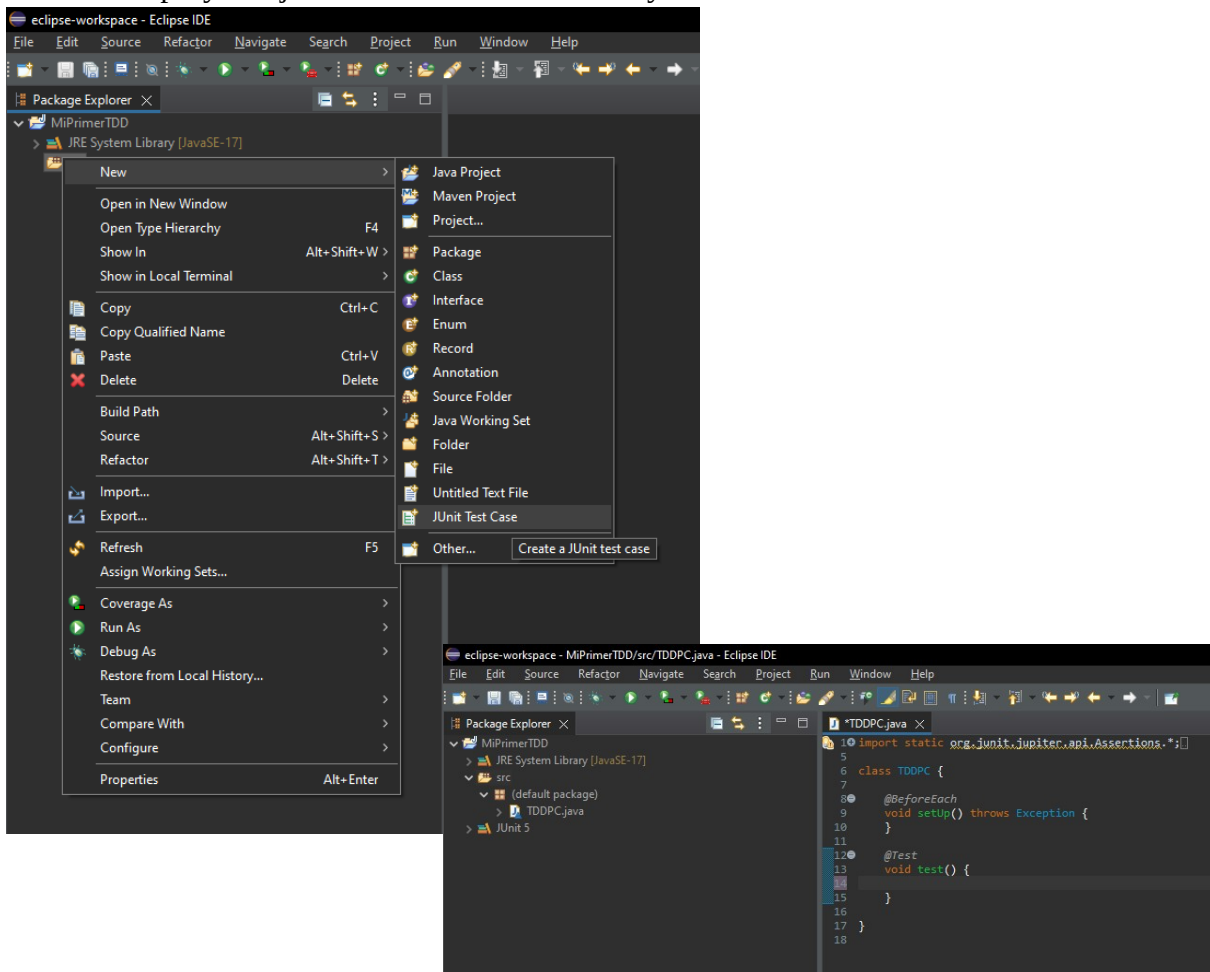
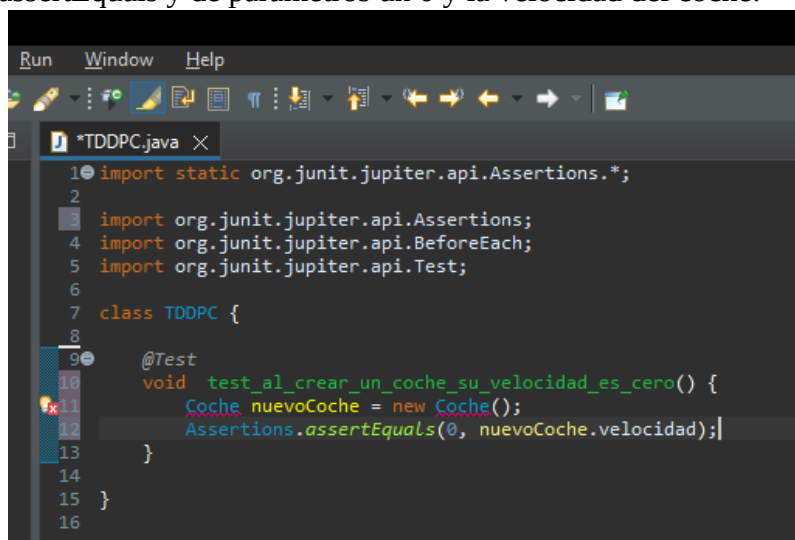


Mi primer TDD

Creamos un proyecto java llamado miPrimerTDD y una Junit Test Case:



Creamos un método y dentro hacemos un objeto Coche e importamos Assertions con el método assertEquals y de parámetros un 0 y la velocidad del coche:



Creamos la class Coche:

```
9  @Test
10 void test_al_crear_un_coche_su_velocidad_es_cero() {
11     Coche nuevoCoche = new Coche();
12 }
13 }
14 }
15 }
16 }
```

Coche cannot be resolved to a type

7 quick fixes available:

- Create class 'Coche'

Creamos el atributo velocidad en la Class Coche para que funcione el assertEquals:

```
velocidad_es_cero() {
();
nuevoCoche.velocidad);|
```

velocidad cannot be resolved or is not a field

2 quick fixes available:

- Create field 'velocidad' in type 'Coche'
- Create constant 'velocidad' in type 'Coche'

Window Help

TDDPC.java Coche.java

```
1
2 public class Coche {
3
4     public int velocidad;
5
```

Testeamos y comprobamos que está todo correcto, el atributo velocidad debe ser un int para que el 0 que hay como parámetro dentro del método assertEquals no dé error:

Finished after 0,191 seconds

Runs: 1/1 Errors: 0 Failures: 0

TDDPC [Runner: JUnit 5] (0,033 s)

test_al_crear_un_coche_su_

Go to File

Run

Debug

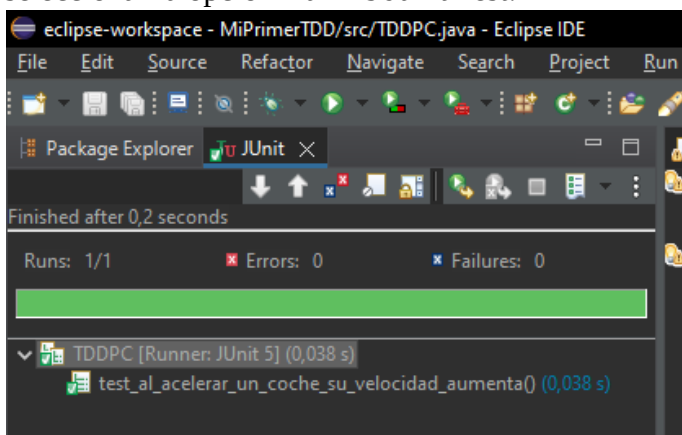
Creamos un nuevo método para acelerar en la class TDDPC:

```
*TDDPC.java Coche.java
1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 class TDDPC {
8
9     @Test
10     void test_al_crear_un_coche_su_velocidad_es_cero() {
11         Coche nuevoCoche = new Coche();
12         Assertions.assertEquals(0, nuevoCoche.velocidad);
13     }
14
15     @Test
16     public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
17         Coche nuevoCoche = new Coche();
18         nuevoCoche.acelerar(30);
19         Assertions.assertEquals(30, nuevoCoche.velocidad);
20     }
21 }
22 }
```

Y para ese método, creamos otro llamado `acelerar` en la class `Coche`:

```
TDDPC.java  *Coche.java x
1
2 public class Coche {
3
4     public int velocidad;
5
6     public void acelerar(int aceleracion) {
7         velocidad += aceleracion;
8     }
9
10
11
12 }
13
```

Testeamos este método y todo correcto, debemos clicar en cada método con el botón izq y seleccionar la opción `Run As JUnit Test`:



Creamos un método para decelerar en `TDDPC`:

```

1 import static org.junit.jupiter.api.Assertions.*;
2
3 import org.junit.jupiter.api.Assertions;
4 import org.junit.jupiter.api.BeforeEach;
5 import org.junit.jupiter.api.Test;
6
7 class TDDPC {
8
9     @Test
10     void test_al_crear_un_coche_su_velocidad_es_cero() {
11         Coche nuevoCoche = new Coche();
12         Assertions.assertEquals(0, nuevoCoche.velocidad);
13     }
14
15     @Test
16     public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
17         Coche nuevoCoche = new Coche();
18         nuevoCoche.acelerar(30);
19         Assertions.assertEquals(30, nuevoCoche.velocidad);
20     }
21
22     @Test
23     public void test_al_decelerar_un_coche_su_velocidad_disminuye() {
24         Coche nuevoCoche = new Coche();
25         nuevoCoche.velocidad = 50;
26         nuevoCoche.decelerar(20);
27         Assertions.assertEquals(30, nuevoCoche.velocidad);
28     }
29 }

```

A su vez, creamos el método decelerar en class Coche:

The screenshot shows the Eclipse IDE with two files open: `TDDPC.java` and `Coche.java`. In `TDDPC.java`, a JUnit test is being written. A red squiggly line under the `decelerar(20)` call triggers an error message: "The method decelerar(int) is undefined for the type Coche". A tooltip shows three quick fixes: "Change to 'acelerar(...)'", "Create method 'decelerar(int)' in type 'Coche'", and "Add cast to 'nuevoCoche'". The `Coche.java` file shows the implementation of the `Coche` class with a `velocidad` attribute and `acelerar` and `decelerar` methods.

```
@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);
}
```

```
public class Coche {
    public int velocidad;

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

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

Testeamos y todo correcto, cada vez que testeamos tenemos que hacer con el método concreto que queremos testear:

The screenshot shows the Eclipse IDE with the JUnit test runner. The test runner shows that the test `test_al_decelerar_un_coche_su_velocidad_disminuye()` passed successfully. The test runner also shows the test results for the other two tests: `test_al_acelerar_un_coche_su_velocidad_aumenta()` and `test_al_crear_un_coche_su_velocidad_es_cero()`.

Creamos un método más en TDDPC para controlar las velocidades negativas, que no existen:

The screenshot shows the `TDDPC.java` file with the following code:

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

class TDDPC {

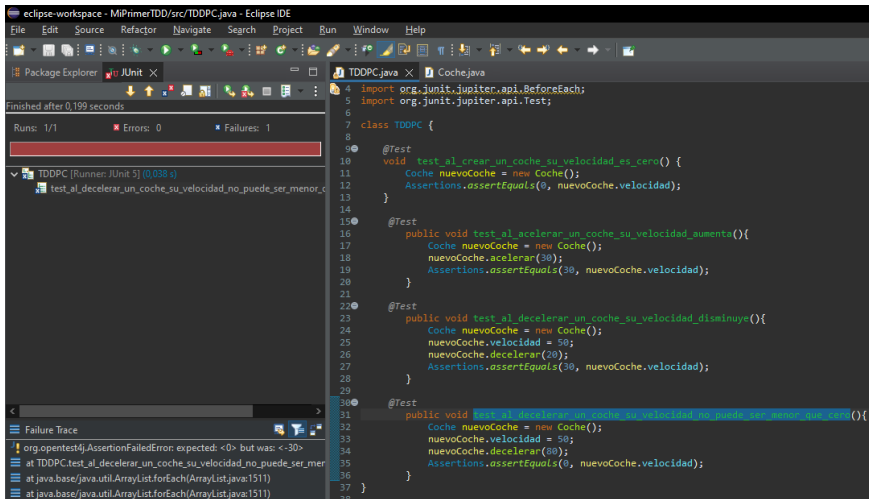
    @Test
    void test_al_crear_un_coche_su_velocidad_es_cero() {
        Coche nuevoCoche = new Coche();
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }

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

    @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);
    }

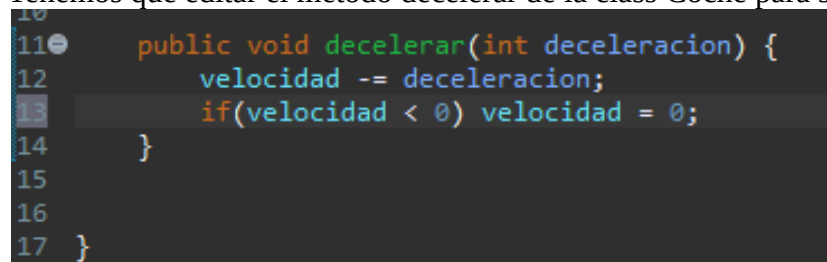
    @Test
    public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
        Coche nuevoCoche = new Coche();
        nuevoCoche.velocidad = 50;
        nuevoCoche.decelerar(80);
        Assertions.assertEquals(0, nuevoCoche.velocidad);
    }
}
```

al testear este método nos da error!!



Eso es porque no podemos manejar valores negativos, si nos fijamos siempre hemos manejado valores positivos, porque la velocidad nunca puede ser menor que 0, pero si deceleramos en 80 teniendo como velocidad 50, sale negativo, y assertEquals no lo maneja.

Tenemos que editar el método decelerar de la class Coche para solucionar esto:



Ahora testeamos y si está todo correcto:

