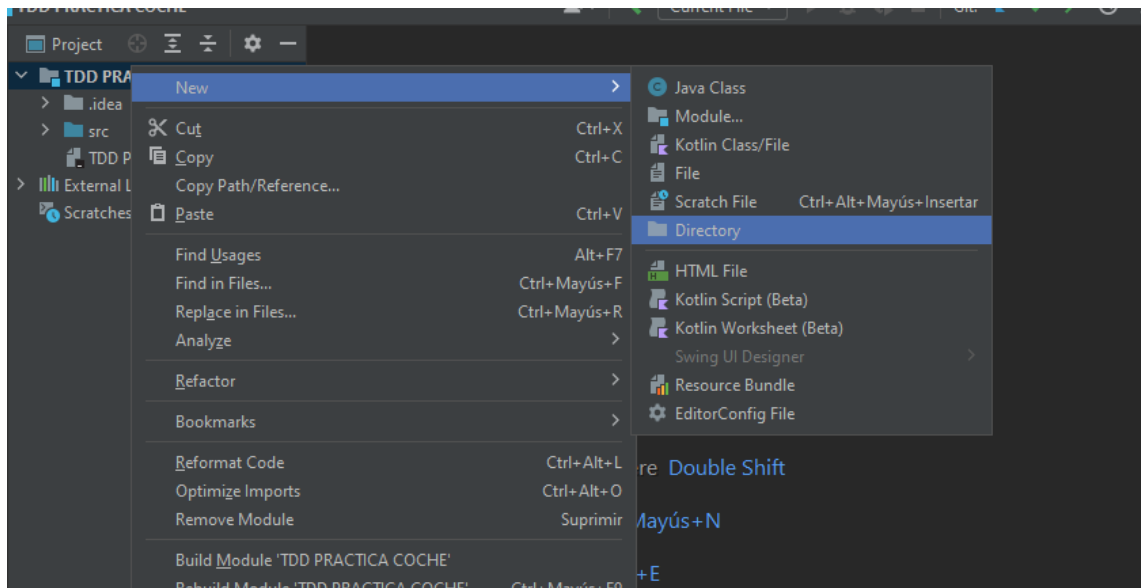
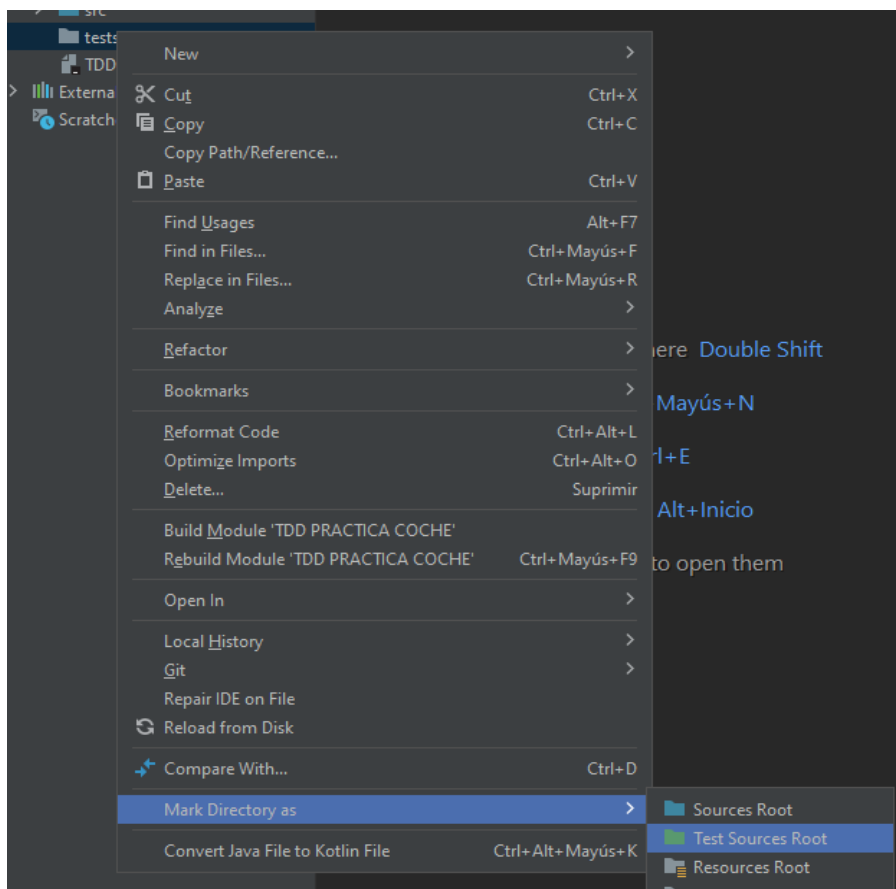


Mi primer TDD

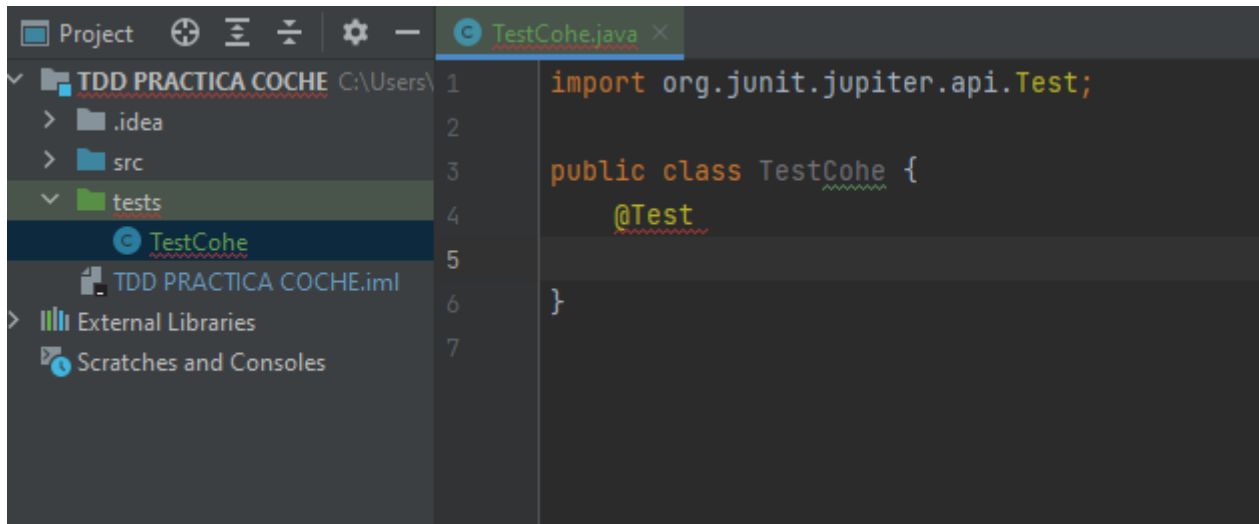
Lo primero que hacemos es crear un nuevo directorio que se llame tests



Después lo marcamos como un directorio de tests



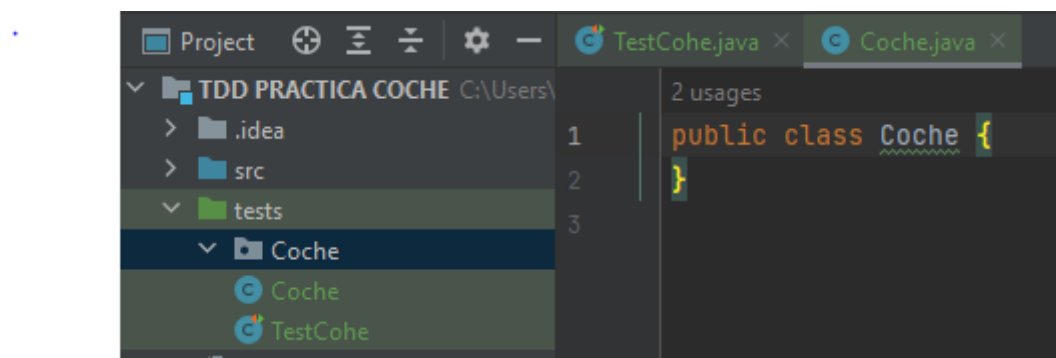
Creamos una clase e indicamos que es un test



The screenshot shows the IntelliJ IDEA interface. On the left, the 'Project' view displays the directory structure: 'TDD PRACTICA COCHE' (containing '.idea', 'src', and 'tests'), 'TDD PRACTICA COCHE.iml', 'External Libraries', and 'Scratches and Consoles'. The 'tests' folder is expanded, and 'TestCohe' is selected. The main editor window shows the code for 'TestCohe.java':

```
1 import org.junit.jupiter.api.Test;
2
3 public class TestCohe {
4     @Test
5
6 }
7
```

Creamos la clase Coche

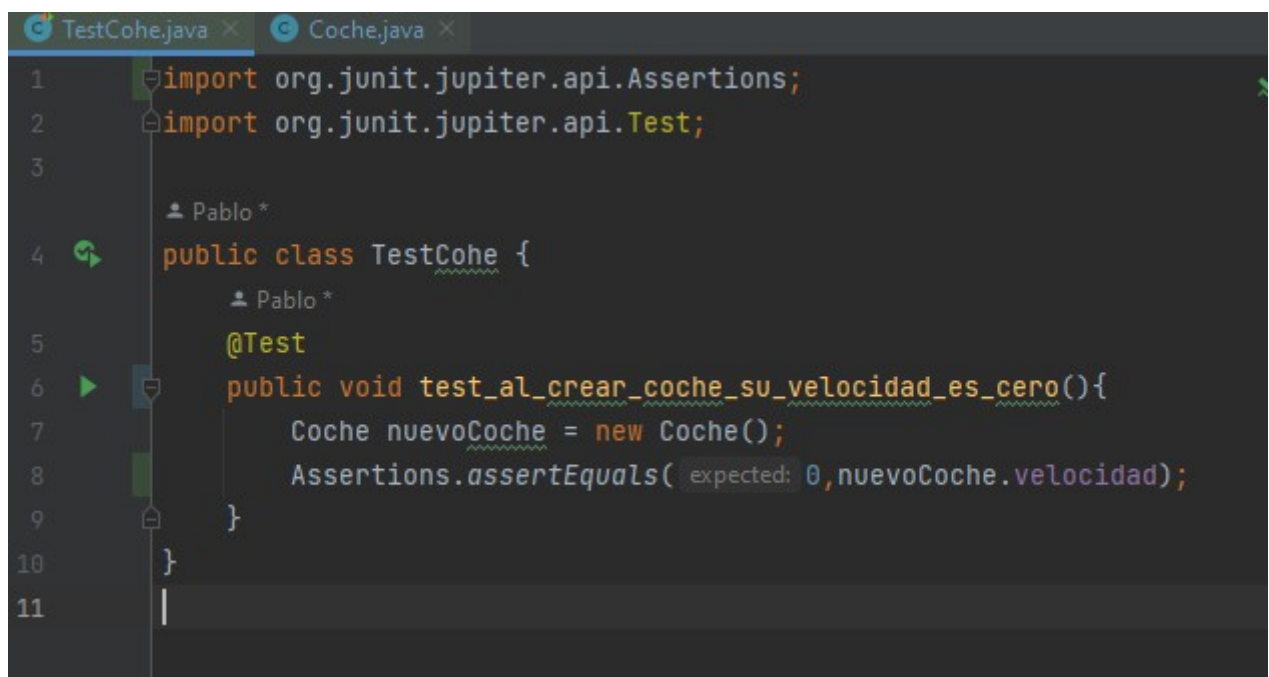


The screenshot shows the IntelliJ IDEA interface. The 'Project' view on the left shows the 'Coche' folder selected under 'tests'. The main editor window shows the code for 'Coche.java':

```
1 public class Coche {
2
3 }
```

Below the code, it indicates '2 usages'.

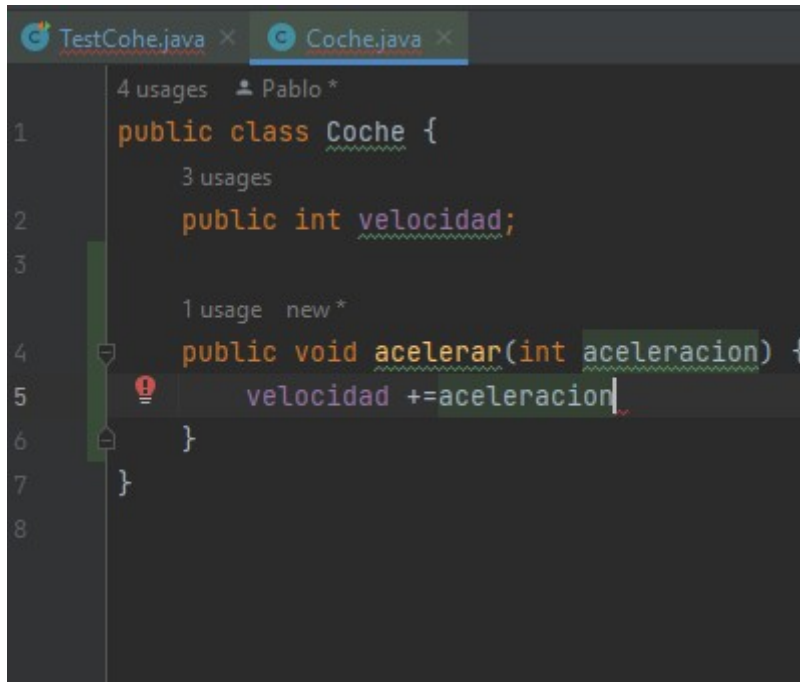
Como el primer test se ejecuto con éxito complicamos un poco el código



The screenshot shows the IntelliJ IDEA interface with two tabs: 'TestCohe.java' and 'Coche.java'. The 'TestCohe.java' tab is active, showing the updated code:

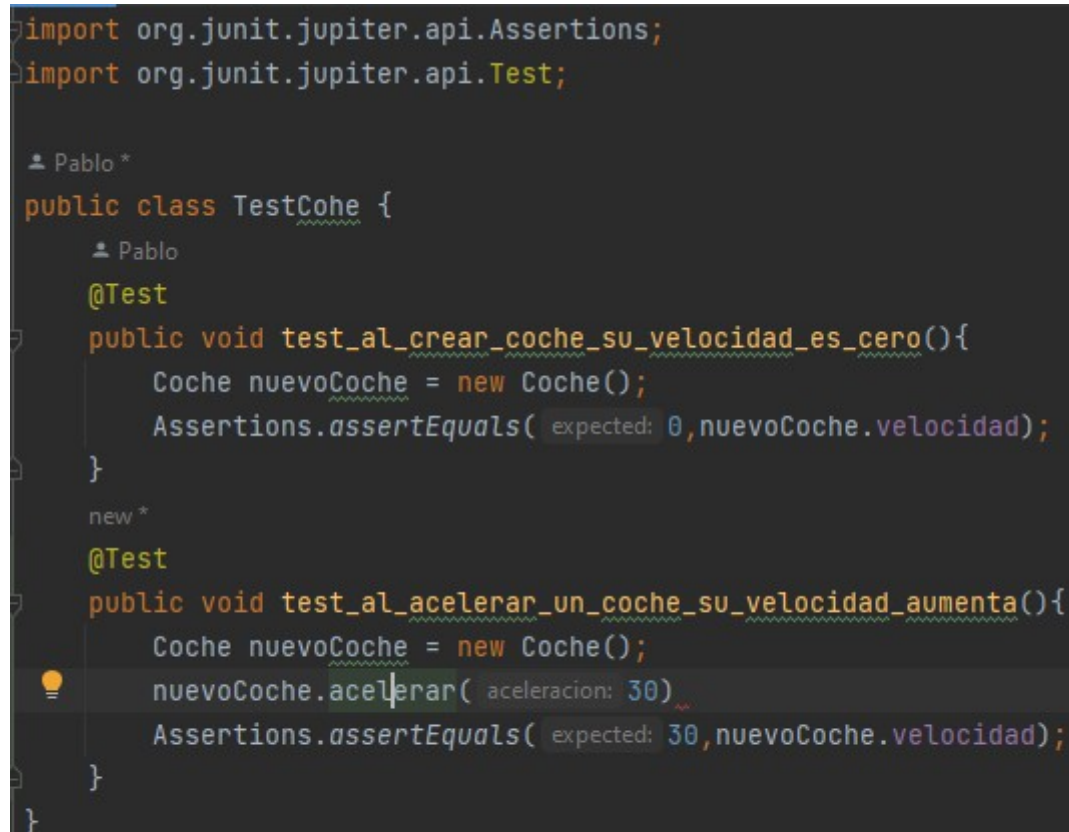
```
1 import org.junit.jupiter.api.Assertions;
2 import org.junit.jupiter.api.Test;
3
4 public class TestCohe {
5     @Test
6     public void test_al_crear_coche_su_velocidad_es_cero(){
7         Coche nuevoCoche = new Coche();
8         Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
9     }
10 }
11
```

Creamos en coche el metodo acelerar con la entero aceleracion



```
1 public class Coche {
2     public int velocidad;
3
4     public void acelerar(int aceleracion) {
5         velocidad += aceleracion;
6     }
7 }
8
```

Modificamos el codigo y añadimos aceleracion junto con un nuevo metodo



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

public class TestCoche {

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

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

Añadimos otro metodo mas con su respectiva variable que es decelerar

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

Creamos este otro metodo pero nos da error debido a que nos da un numero negativo

```
new *
@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);
}
}
```

Para solucionar este error en el metodo de deceleracion ponemos lo siguiente

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