Entornos de Desarrollo Tema 3 Tarea 1

Índice

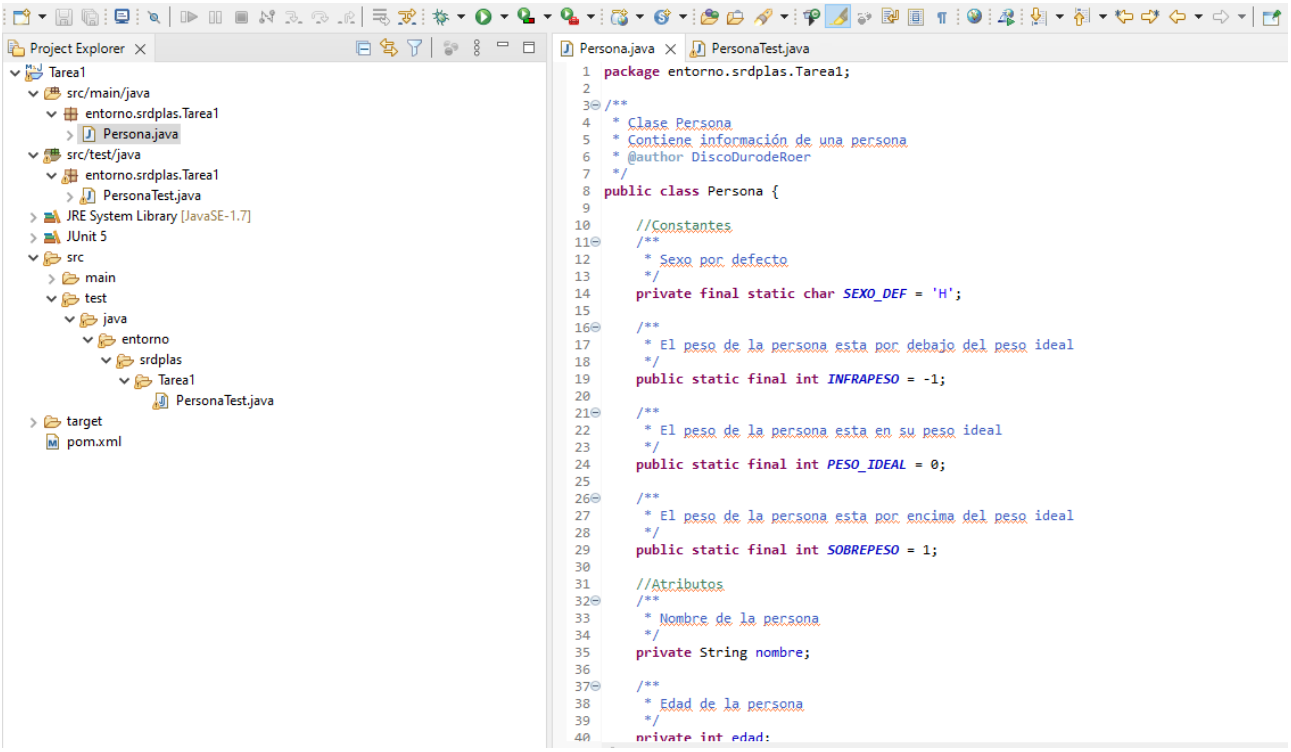
Crear proyecto persona, copiar la clase Persona del enunciado

Importar las librerías de Junit 5 y editar pom.xml

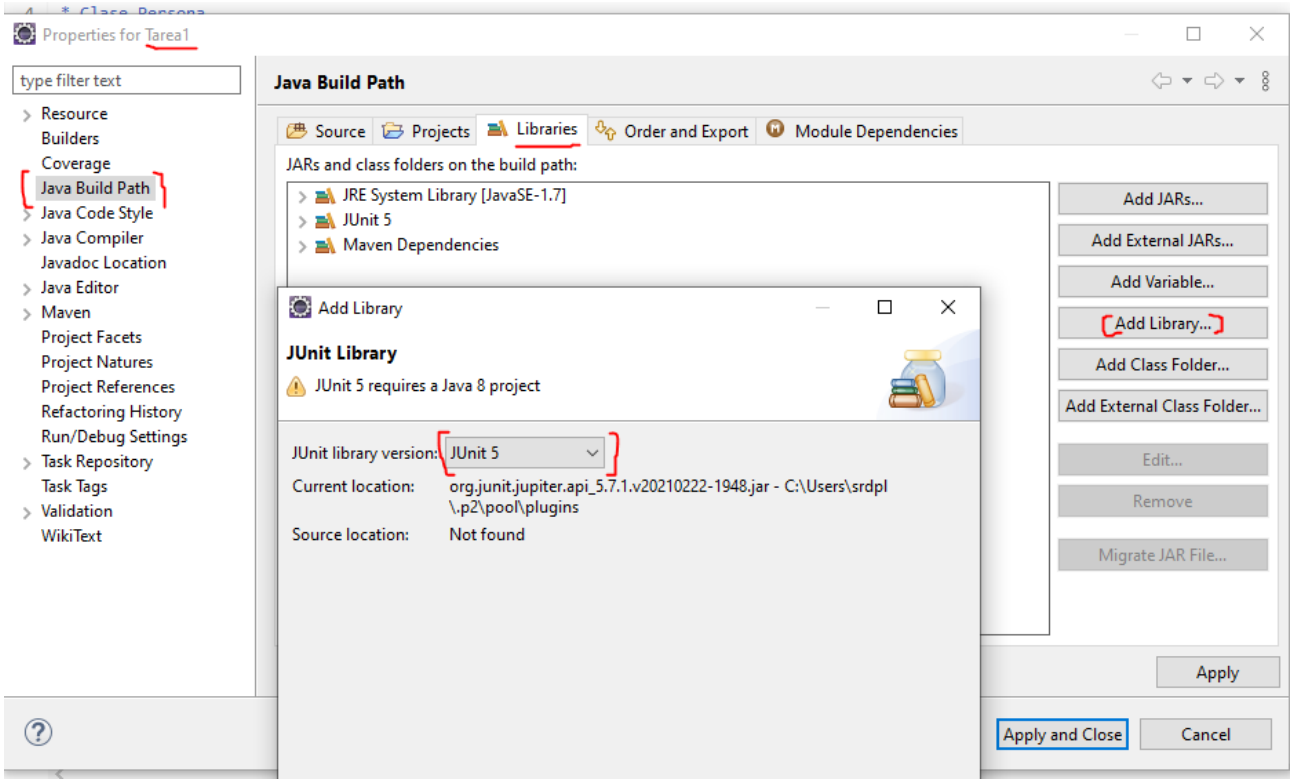
Crear un Test con Junit

Pruebas unitarias a la clase Persona.

Hemos Creado un nuevo proyecto , en el que hemos copiado la clase persona de la tarea 1 de testing.



Importamos las librerías de Junit al proyecto y editamos el pom.xml



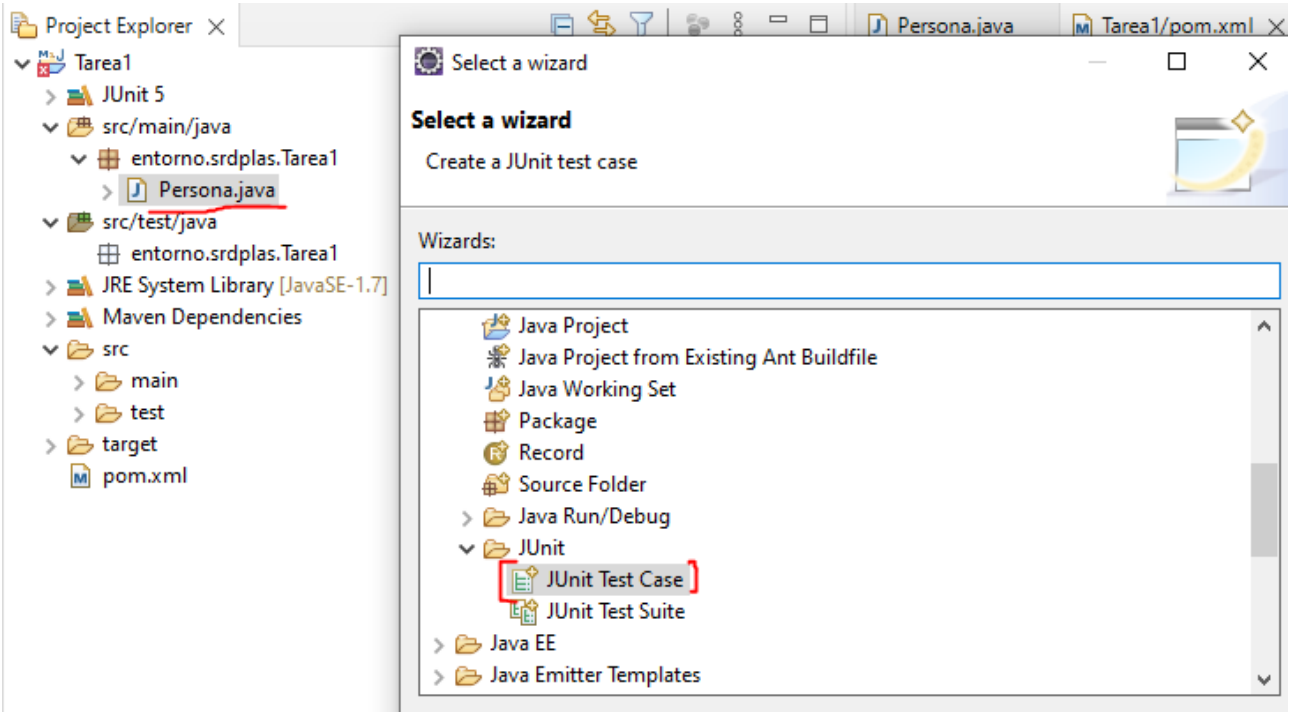

```xml
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <junit-jupiter.version>5.5.2</junit-jupiter.version>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

Cambiamos el maven compiler source a la 1.8 porque a partir de esta versión es con la que funciona el Junit 5

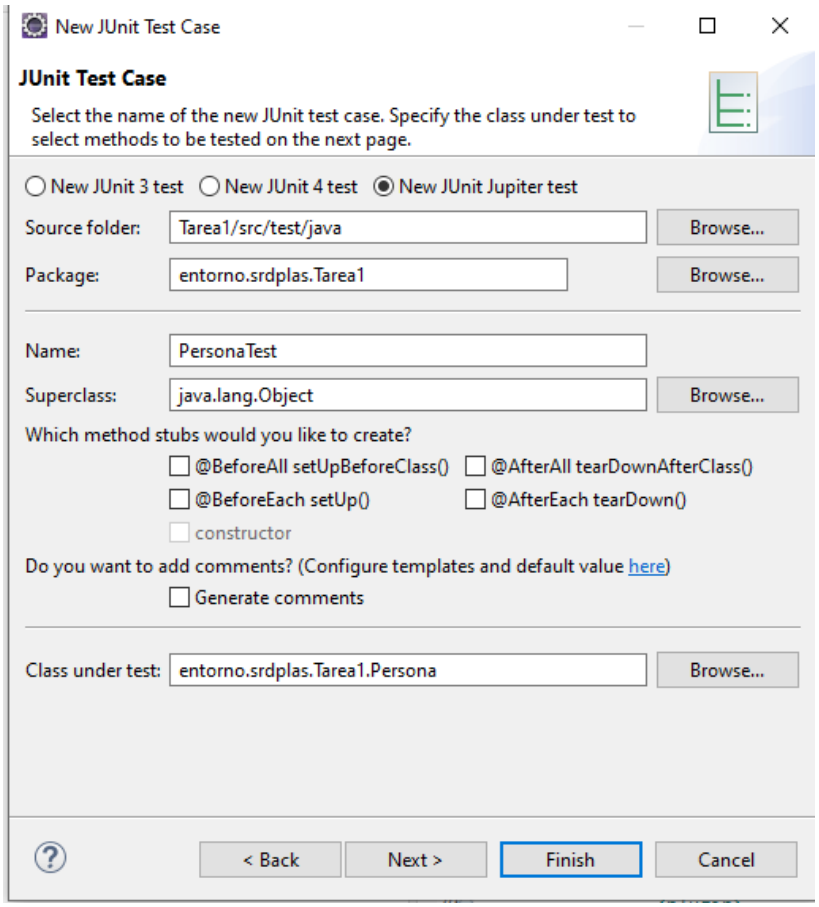
```xml
<dependencies>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.5.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Añadimos el Junit de jupiter que es la version 5, y añadimos las dependencias a este, previamente tendremos que actualizar los plugins maven compiler y el maven surefire

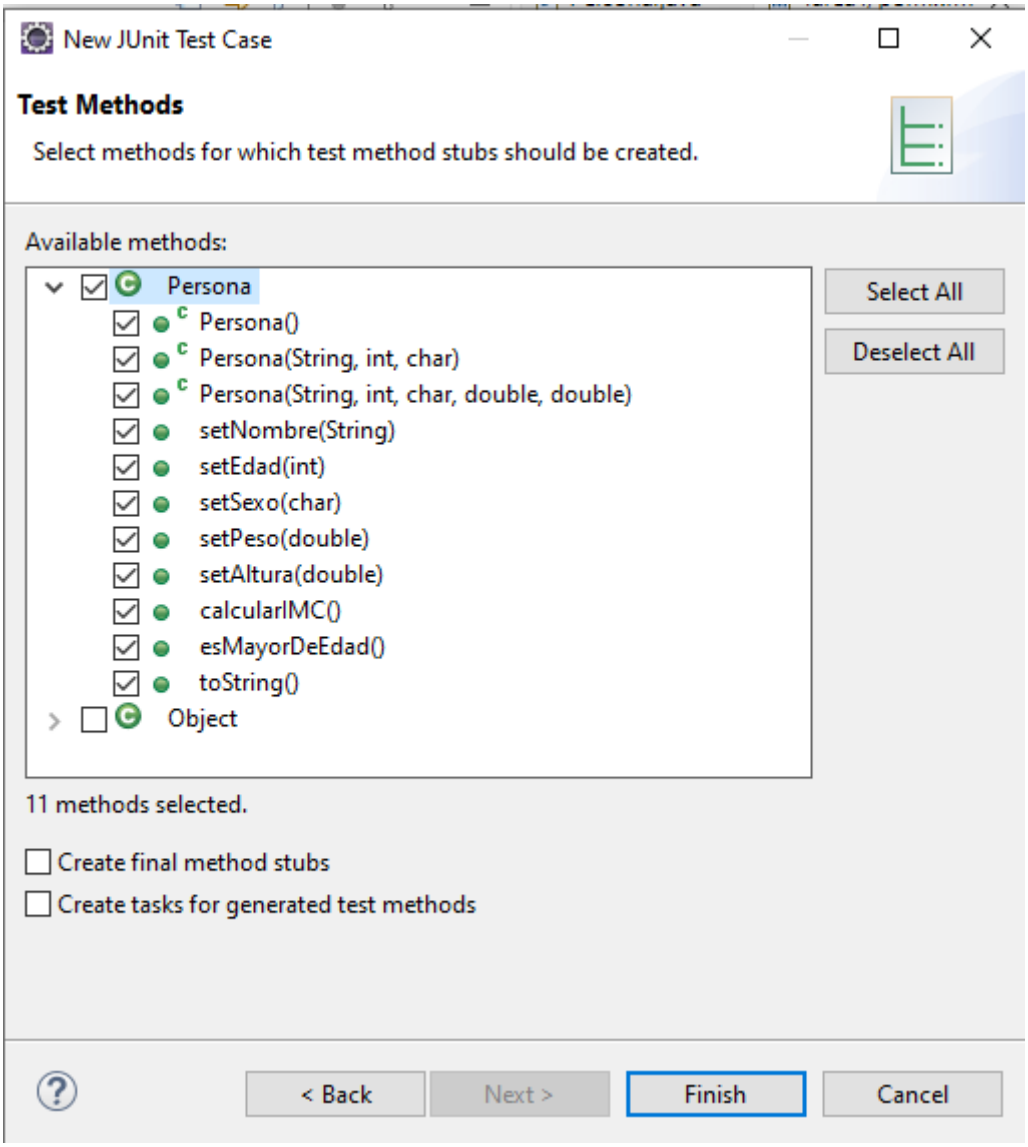
Creamos un test con Junit 5 de la clase Persona.



A continuación tenemos una ventana de configuración del caso de Junit si le damos a finish nos creará el test vacio pero si le damos a next podremos elegir sobre que propiedades quieres hacer un test.



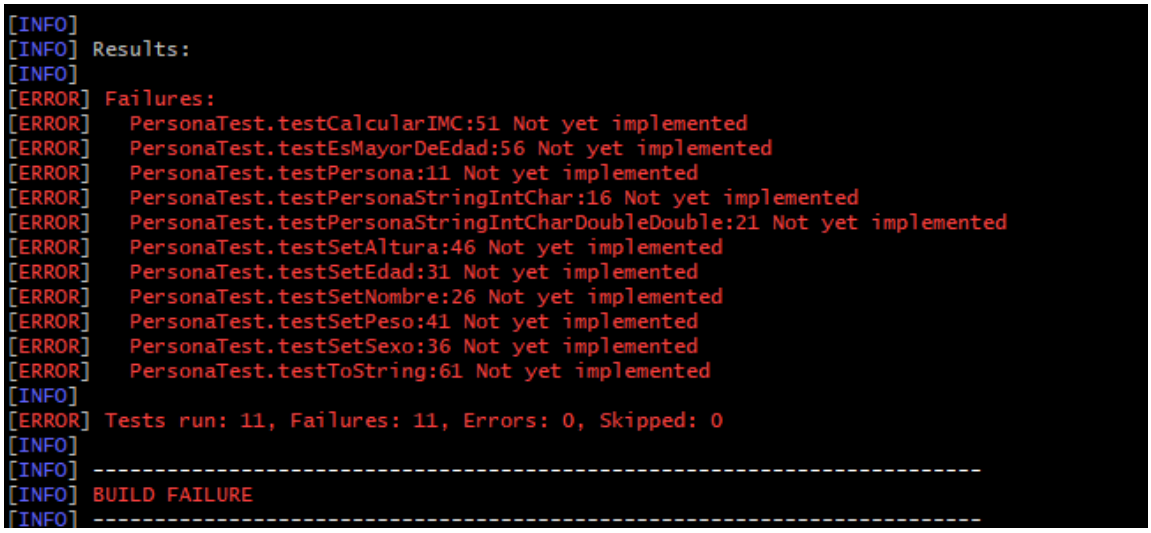
Nosotros le vamos a dar a next para elegir a que le vamos a hacer el test



Vamos a elegir todas los métodos de la clase Persona

```
package entorno.srdplas.Tarea1;

import static org.junit.jupiter.api.Assertions.*;

class PersonaTest {

    @Test
    void testPersona() {
        fail("Not yet implemented");
    }

    @Test
    void testPersonaStringIntChar() {
        fail("Not yet implemented");
    }
```

Si intentamos ejecutar un test con el mvn clean test, nos va a da error por todos lados pero porque por defecto el test que hemos creado está hecho para dar error.



Una vez comprobado lo anterior y que los pasos anteriores los hemos cumplido, procederemos a hacer test individuales añadiendo lineas de código.

Solo hemos hecho Test de los métodos para saber si es mayor de edad, su Indice de Masa Corporal(IMC) y del método toString que mostraba todas las propiedades del objeto persona que hayamos creado.

En el método si esmayor de edad solo comprobamos que el resultado que nos devuelve sea verdadero o falso.

En el método para calcular el IMC solo comprobamos que los valores que devuelve estén en el rango de las constantes que indican si es sobrepeso, peso optimo…

El método toString solo comprobamos que el contenido de este método no sea nulo, automáticamente al crear el objeto se le

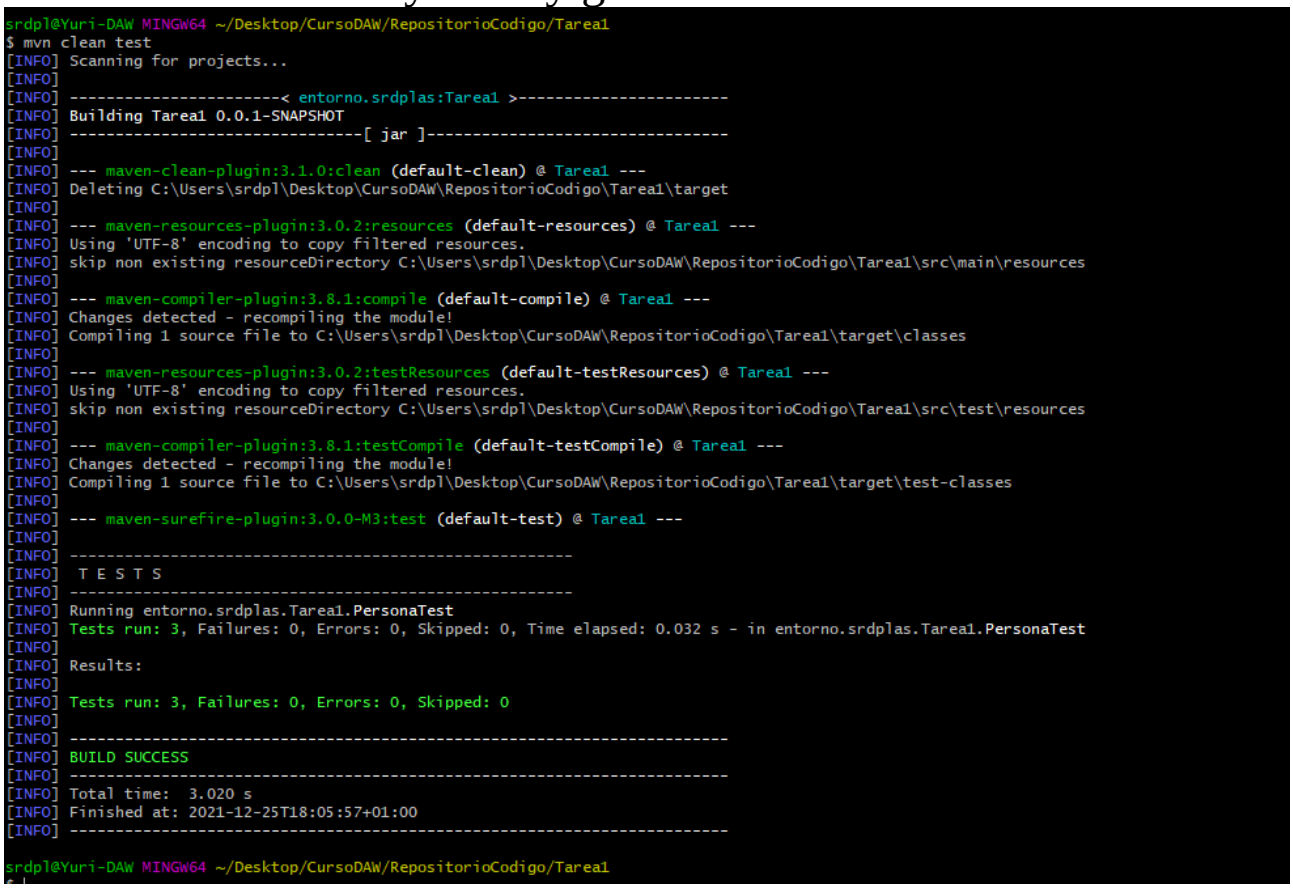
asigna un DNI, por lo tanto con ver que se ha generado y no esté nulo nos valdría ya que no tenemos getters.

PersonaTest.java

```java
package entorno.srdplas.Tarea1;
import static org.junit.jupiter.api.Assertions.*;
import javax.swing.plaf.basic.BasicPanelUI;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.*;
import org.junit.*;
class PersonaTest {

    @Test
    void testCalcularIMC() {
        Persona p = new Persona();
        p.setAltura(1.75);
        p.setEdad(19);
        p.setNombre("Manolo");
        p.setPeso(95.67);

        boolean resultado=false;
        if(p.calcularIMC()>=-1&&1<=p.calcularIMC()) {
            resultado =true;
        }
        // asi contemplamos que el resultado que nos ofrece el metodo esté en los valores
        //Establecidos en las constantes
        assertTrue(resultado=true, "Error resultado no esperado");

    }

    @Test
    void testEsMayorDeEdad() {
        Persona p = new Persona();
        p.setEdad(17);
        boolean funciona = false;
        if(p.esMayorDeEdad()==true||p.esMayorDeEdad()==false) {
            funciona = true;
        }
        //Si el metodo toma los valores true or false, la variable pasará a true
        //El test pasará si esta variable es verdadera
        assertTrue(funciona=true, "Error resultado no esperado");

    }

    @Test
    void testToString() {
        Persona p = new Persona();
        p.setAltura(175);
        p.setEdad(19);
        p.setNombre("Manolo");
        p.setPeso(95.67);

        //Si el contenido del objeto p es nulo no pasará el test
        assertNotEquals(null, p, "Error resultado no esperado");
```

No hemos creado test para los setters porque las propiedades son privadas y no tenemos acceso a esta, también porque la clase persona no tenía ningún get para ver la propiedad.

A los constructores tampoco le hemos dado, si hay algún error ya lo dirá la MainClass y no hay gets.



```java
package entorno.srdplas.Tarea1;
import static org.junit.jupiter.api.Assertions.*;
class PersonaTest {

    private static Persona p;

    @BeforeAll
    static void init() {
        p= new Persona();
    }

    @AfterAll
    static void finish(){
        p = null;
    }


    @Test
    void testCalcularIMC() {

        p.setAltura(1.75);
        p.setEdad(19);
        p.setNombre("Manolo");
        p.setPeso(95.67);

        boolean resultado=false;
        if(p.calcularIMC()>=-1&&1<=p.calcularIMC()) {
            resultado =true;
        }
        // asi contemplamos que el resultado que nos ofrece el metodo esté en los valores
        //Establecidos en las constantes
        assertTrue(resultado=true, "Error resultado no esperado");

    }
```

Este sería el código si usamos el BeforeAll y el AfterAll para iniciar el método init() antes de cada test y el método finish después de cada test.