

JUnit 5

Paso 1

Declaramos esta variable que nos ayudara a realizar operaciones a lo largo de la implementación de los test



```
private CursoService cursoService;
```

Paso 2

@BeforeEach esta notación indica que esto se ejecutara antes de cada prueba

@Test esta notación indica que es una prueba (test)

Aquí realizamos la instancia de nuestro servicio

Utilizamos los comportamientos de nuestro servicio

En resumen el metodo anterior nos evitara hacer instancias en cada test



```
@BeforeEach
@Test
public void setUp(){
    cursoService = new CursoService();
    cursoService.crearCurso(1,"JAVA",1500,"160HRS");
    cursoService.crearCurso(100,"PHP",1200,"180HRS");
    cursoService.actualizarCurso(100,"PHP",1800,"180HRS");
}
```



```
CursoService cursoService = new CursoService();
```

Test 1

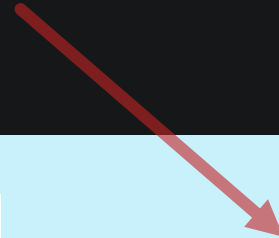
@DisplayName Esta anotación nos ayuda a definir un nombre a nuestro método así como un nombre que describa lo que realizara el test



```
@DisplayName("se espera que el resultado curso sea igual al curso que estamos esperando")
@Test
public void test1(){
    CursoDTO cursoEsperado = new CursoDTO(1,"JAVA", 2500,"160HRS");
    // CursoService cursoService = new CursoService();
    final CursoDTO resultado = cursoService.crearCurso(1,"JAVA", 2500,"160HRS");
    // ES IGUAL QUE OTRO
    Assertions.assertEquals(cursoEsperado,resultado);
}
```



Lo que realiza este test es comprobar si el curso esperado es igual al que llega como resultado por ello ocupamos **assertEquals**, evalúa **true o false**



Si es diferente fallara ejemplo que llegue con un ID diferente del curso esperado

Test 2

Esto que agregamos es para que imprima el problema

```
@DisplayName("Esperamos que el NOMBRE del curso resultaod sea igual al esperado de lo contrario falla")
@Test
public void test2(){
    CursoDTO cursoEsperado = new CursoDTO(1,"JAVA", 2500,"160HRS");
    // CursoService cursoService = new CursoService();
    final CursoDTO resultado = cursoService.crearCurso(1,"JAVA", 2500,"160HRS");
    Assertions.assertEquals(cursoEsperado.getNombreCurso(),resultado.getNombreCurso(),"El nombre del curso es diferente");
}
```

Lo que realiza este test es comprobar si el nombre del curso esperado es igual al nombre del curso de resultado

Verifica mediante los atributos (estados) de nuestra clase

Test 3

Este ejemplo podría ser la búsqueda de un id de una base de datos si es realmente el id que buscamos o es diferente

```
@DisplayName("Esperamos que el ID del curso resultado sea igual al del esperado")
@Test
public void test3(){
    CursoDTO cursoEsperado = new CursoDTO(1,"JAVA", 2500,"160HRS");
    // CursoService cursoService = new CursoService();
    final CursoDTO resultado = cursoService.crearCurso(1,"PHP", 2500,"160HRS");
    Assertions.assertEquals(cursoEsperado.getIdCurso(),resultado.getIdCurso());
}
```

Lo que realiza este test es comprobar si el id del curso esperado es igual al id del curso de resultado

Verifica mediante el id (estados) de nuestra clase

Test 4

Este ejemplo podría ser la búsqueda de un id de una base de datos

```
@DisplayName("OBTENER EL CURSO QUE BUSCAMOS POR ID")
@Test
public void test4(){
    CursoDTO cursoEsperado = new CursoDTO(100,"PHP",1500,"160hrs");
    // CursoService cursoService = new CursoService();
    // CREAMOS EL CURSO
    // cursoService.crearCurso(100,"PHP",1500,"160HRS");
    final CursoDTO resultado = cursoService.obtenerCurso(100);
    Assertions.assertEquals(cursoEsperado.getIdCurso(),resultado.getIdCurso());
}
```

Lo que realiza este test es comprobar si el id del curso esperado es igual al id del curso de resultado

Verifica mediante el id (estados) de nuestra clase si realmente es el registro que buscamos

Test 5

Este ejemplo podría ser la actualización de un id de una base de datos

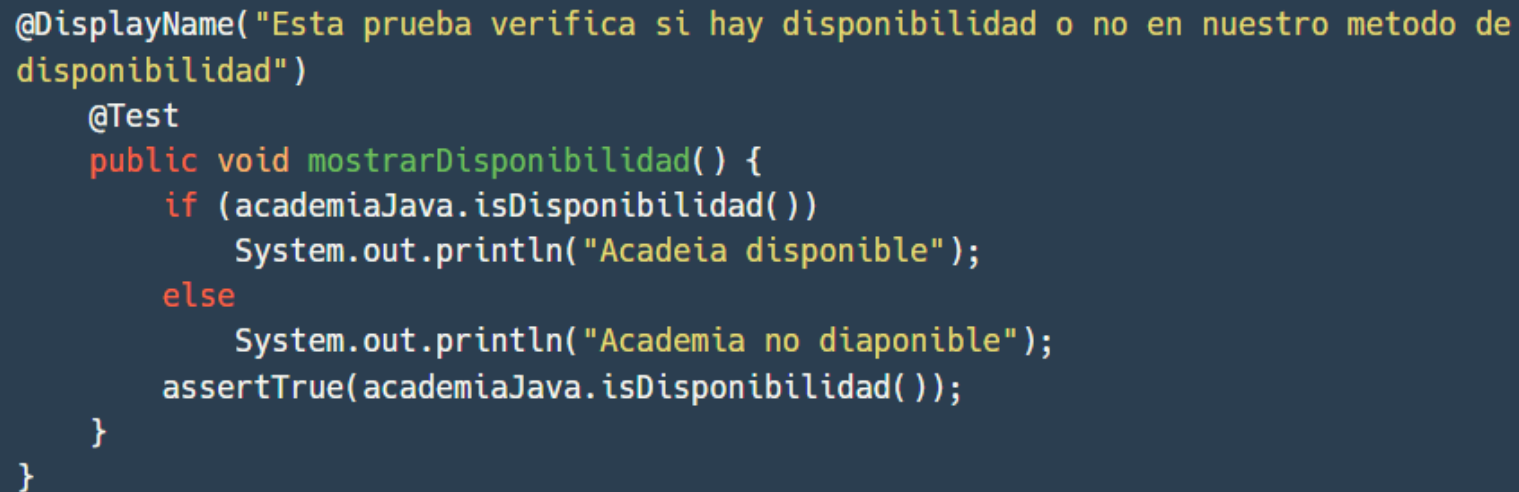
```
@DisplayName("ACTUALIZAR EL CURSO POR ID")
@Test
public void test5(){
    CursoDTO cursoEsperado = new CursoDTO(100,"PHP",1800,"180hrs");
    // CursoService cursoService = new CursoService();

    // CREAMOS EL CURSO
    // cursoService.crearCurso(100,"PHP",1500,"160HRS");
    final CursoDTO resultado = cursoService.actualizarCurso(100,"PHP",1800,"180HRS");
    Assertions.assertEquals(cursoEsperado.getIdCurso(),resultado.getIdCurso());
}
```

Lo que realiza este test es actualizar un curso siempre y cuando sea igual al id que se desea actualizar

Compara los id y si son iguales pasa el test

En este test nos ayuda a verificar si hay disponibilidad



```
@DisplayName("Esta prueba verifica si hay disponibilidad o no en nuestro metodo de disponibilidad")
@Test
public void mostrarDisponibilidad() {
    if (academiaJava.isDisponibilidad())
        System.out.println("Acadeia disponible");
    else
        System.out.println("Academia no diaponible");
    assertTrue(academiaJava.isDisponibilidad());
}
```

Nos tiene que regresar true


Nos ayuda a setear un nombre de academia pero tambien hacemos una comparación

Tapia-Dev

```
@DisplayName("Nos ayuda a identificar si el nombre setado es igual al que esperamos")
@Test
public void llenarDatos(){
    academiaJava.setNombreAcademia("Java");
    assertEquals(expected: "Java", academiaJava.getNombreAcademia(), message: "El nombre NO es igual NO pasa el test")
}
```

Aquí realizar la comparación del nombre esperado y el que llega

Este test nos ayuda a verificar que no existan menos de 10 alumnos inscritos



```
Tapia-Dev
@Test
void verificarAlumnos() {
    academiaJava.setAlumnosAcademia(15);
    assertFalse(condition: academiaJava.getAlumnosAcademia()<10, message: "Hay menos de 10 alumnos " +
        "no puedes iniciar la academia");
}
```



Aquí realizar la comparación del nombre esperado y el que llega