Auditoria, Calidad y Gestion de Sistemas
(ACG)

Práctica Voluntaria
Testeando un Reloj

con mas JUnit Tests

Curso 22/23

Esta práctica es voluntaria y no cuenta para la nota de observación. Cubre algunos aspectos adicionales como los test paramétricos o las excepciones.

1. Time

Descarga el archivo *Time.java* y cópialo en el directorio src del proyecto. Esta clase es básicamente una representación simple de una hora del día, utilizando el reloj de 24 horas. Para probar el comportamiento de la clase Time usando JUnit, debes crear una clase (*TimeTest.java*) separada que contenga todas las pruebas. Crea dicha clase usando la opción "New JUnit Test Case".

2. Creando un Test Sencillo

Añade la siguiente funcion de test a *TimeTest.java*:

```
1  @Test
2  public void testGetHours() {
3      Time midnight = new Time(0, 0, 0);
4      Time noon = new Time(12, 0, 0);
5      Time elevenPm = new Time(23, 0, 0);
6
7      assertEquals(midnight.getHours(),0);
8      assertEquals(noon.getHours(), 12);
9      assertEquals(elevenPm.getHours(), 23);
10 }
```

Comprueba que funciona correctamente.

3. Tests Paramétricos

El método set se utiliza para cambiar la hora del reloj. Devuelve verdadero cuando el cambio es posible y falso en caso contrario. Un método de test para este metodo debe comprobar tanto valores válidos como inválidos para las horas, minutos y segundos:

```
1  @Test
2  public void setTest() {
3      Time t=new Time(1, 0, 0);
4      assertTrue(t.set(1,0,0));
5      assertTrue(t.set(0,0,0));
6      assertTrue(t.set(24,0,0));
7      assertFalse(t.set(25,0,0));
8      assertFalse(t.set(-1,0,0));
9      ...
10     }
```

En estos casos donde hay muchos valores a probar puede ser mas eficiente usar un test paramétrico[1] como el siguiente:

```
@ParameterizedTest
@CsvSource({"1, 0, 0,true","0,0,0,true","24,0,0,true","
    25,0,0,false","-1,0,0,false"})
public void setTest(int hours,int minutes, int seconds,
    boolean expected) {
    Time t=new Time(1, 0, 0);
    assertEquals(t.set(hours,minutes,seconds),expected);
    }
```

Completa el test anterior para testear los valores de los minutos y segundos. Ejecuta sus pruebas y, si fallan, corrige los problemas correspondientes en la clase *Time.java*.

4. Testeando Excepciones

Hasta ahora, solo hemos probado que el objeto de tiempo se inicializa correctamente cuando se proporciona un valor legal al constructor. ¿Qué pasa si en su lugar se proporciona un valor ilegal? Los comentarios de la clase *Time*, dichos valores deberían desencadenar un *IllegalArgumentException*, por lo que debes verificar que esto es lo que realmente sucede.

Para ello añade la siguiente función a *TimeTest.java*:

```
@Test
public void constructorException() {
    try {
       new Time(-1, 0, 0);
       fail("IllegalArgumentException expected.");
       } catch (IllegalArgumentException expected){
          assertEquals("hours out of range", expected.
             getMessage());
       }
}
```

5. Tareas

1. Crea un test paramétrico para comprobar que el constructor devuelve todas las excepciones correctamente. Ejecuta sus pruebas y, si fallan, corrige los problemas correspondientes en la clase *Time.java*.

2. Estudia el método equals en la clase Time. Esto debería devolver: verdadero cuando un objeto de tiempo se compara consigo mismo;

[1] Puedes leer más sobre test paramétricos aquí `https://www.baeldung.com/parameterized-tests-junit-5`

verdadero cuando se compara con un objeto de tiempo diferente que tiene el mismo estado interno; falso cuando se compara con un objeto de tiempo diferente que tiene un estado interno diferente; falso cuando se compara con algún otro tipo de objeto (independientemente de si ese otro objeto se parece al objeto Tiempo de alguna manera). Crea un método para testearlo,

3. Añade pruebas para los métodos *inSeconds* y *plus* de *Time.java*. Recuerda verificar que plus genere una IllegalArgumentException cuando se espera que lo haga.

   Ejecuta sus pruebas y, si fallan, corrige los problemas correspondientes en la clase *Time.java*.

4. Si has reutilizado los mismos objetos de tiempo en varias pruebas, refactoriza *TimeTest.java* para que estos objetos se creen en un método se setup. ¡Asegúrate de que las pruebas aún se comporten correctamente después!