Laboratorio S6
JUnit 5.0

Objetivos:

- Entender la necesidad de verificar el software
- Sistematizar pruebas unitarias en Java con JUnit 5

Herramientas que vamos a utilizar:

- Entorno de desarrollo Eclipse
- Librería Junit

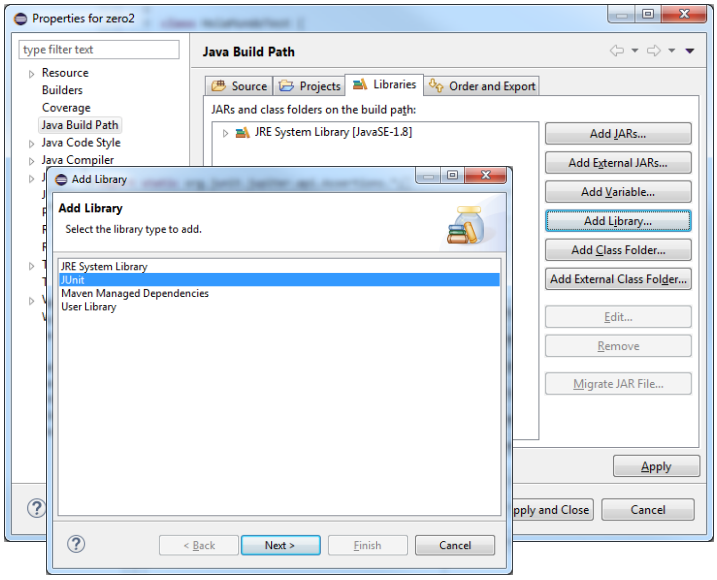
Tareas a realizar

1ª Parte

1) Importa el proyecto `PROYCounter` que tienes en eGela. Este proyecto tiene un paquete *packUnit* donde está la clase `Counter` que vamos a verificar con JUnit.

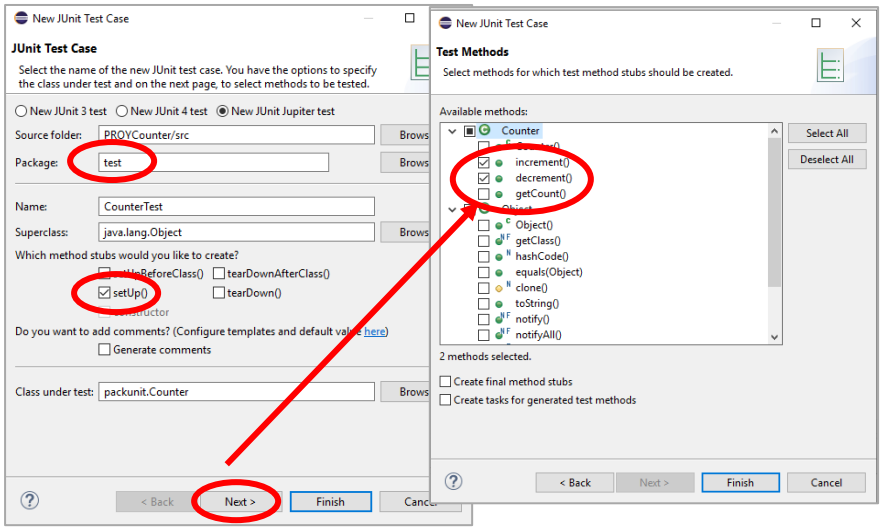
2) Crearemos un package **`test`** donde crearemos los test para verificar la clase.

3) Añade la librería de JUnit al path: ir a *Project→Properties* y en la opción Java Build Path activa la pestaña Libraries, pulsa la opción Add Library y selecciona la librería JUnit (la versión 5)



4) Para generar la clase de test para la clase **`Counter`**, sitúate sobre la misma (selecciónala), pulsa botón derecho y ejecuta *New →JUnit Test Case* (si no aparece la opción selecciona *Other* y busca la carpeta Java y dentro JUnit, y localiza la opción *JUnit Test Case*). Sigue los pasos marcados a continuación:

- Observa cómo se llama la nueva clase: **`CounterTest`**
- Indica que el `package` es **`test`**, marca para que cree el método `setUp()` y pulsa `Next`.
- Selecciona los métodos de la clase que quieres probar: `increment` y `decrement`.



5) En la clase test creada **`CounterTest`** copia lo siguiente:

- Declara el atributo counter1 (realiza la importación necesaria) en la clase:

```
private Counter counter1;
```

- **setUp,** crea un objeto nuevo para cada test que se ejecute:

```
// Generate the objects to be used in each test
counter1 = new Counter();
```

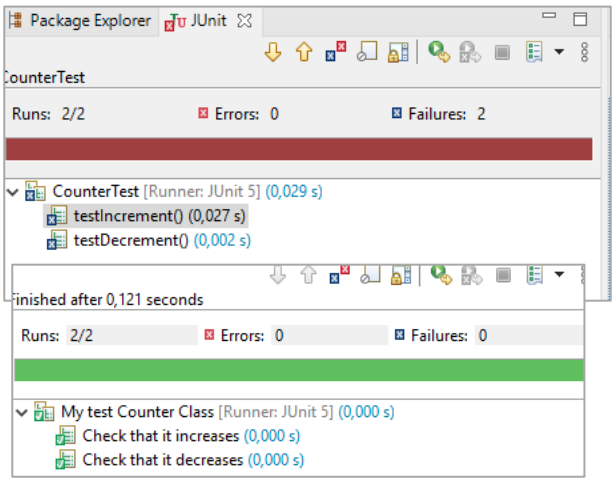
- **`testIncrement`**, incrementa una vez el contador y comprueba que la segunda vez que lo incrementa su valor es 2:

```
counter1.increment();
assertEquals(2, counter1.increment());
```

- **`testDecrement`**, decrementa una vez el contador y comprueba que el valor devuelto es -1:

```
assertTrue(counter1.decrement() == -1);
```

6) Ejecuta los Test, para ello *Run→Run as→JUnitTest*. O ponte sobre la clase test y con el botón derecho selecciona *Run as→→JUnitTest*. O poniéndote sobre el proyecto botón derecho selecciona *Run as→JUnitTes*



7) Analiza qué ocurre.

8) Cambia la constructora de la clase `Counter` e inicializa el atributo `count` a 10. Vuelve a ejecutar y mira lo que sucede.

9) Utiliza la anotación @DisplayName:

   a. Para la clase por ejemplo: @DisplayName("My Test Class")

   b. Para cada test, por ejemplo: @DisplayName("`Check that it increases`")

   c. Vuelve a ejecutar y mira lo que sucede.

2ª Parte[1]

10) Importa el proyecto `NewMailSystem` que está disponible en eGela. Contiene una versión parcial del proyecto del servidor de correo con el que hemos estado trabajando en los laboratorios anteriores. Esta versión tiene menos funcionalidades y alguna distinta.

11) Verifica el método **`equalsReceiver`**.

Para la implementación:

a. Define un atributo `Email` en la clase e inicialízalo en el setUp:

```
@BeforeEach
public void setUp() {
// Generate the objects to be used in all the test methods
   email1=new Email("Ainhoa", "Yeray","Weekend plan");
}
```

b. Este Email servirá para realizar las comparaciones en los dos casos de pruebas siguientes:

- Cuando coinciden los receptores. Implementa este caso en un método dentro de la clase de pruebas con dos mensajes que tengan el mismo `receiver`.
- Cuando no coinciden. Implementa este caso en otro método con dos mensajes que no tengan el mismo `receiver.`
- Implementa cada uno de los casos en un método de test diferente

12) Verifica el método **`storeEmail`**.

a. Define los atributos Email, EmailServer y una lista, para ser utilizados en el SetUp (mira más abajo). Declara adecuadamente los atributos referenciados para la clase `test: email1, email2, mServer y list.` La lista nos servirá para realizar las comprobaciones sobre el estado del servidor.

b. Copia en la clase test para `storeEmail` el método **`setUp`** que se te proporciona a continuación

```
@BeforeEach
public void setUp() {
// Generate the objects to be used in all the test methods
      email1=new Email("Ainhoa", "Yeray","Weekend plan");
      email2=new Email("Yaiza", "Yeray","Meeting Schedule");
      mServer= new MailServer();
      list= new ArrayList<Email>();
}
```

c. Casos de prueba a considerar:

- Estando vacío el servidor, almacena uno.
- Teniendo en el servidor ya un mensaje almacenado, almacena otro.
- Implementa los dos casos de prueba comentados. No hagas uso del método **`getEmailList`** para su verificación.

[1] Recuerda que para poder utilizar `assertEquals` entre objetos de tipo **`Email`** y que funcione adecuadamente debe estar sobrescrito en **`Email`** el método `equals`[1] `(de Object)`. Para ello considera que dos emails son iguales si todos sus atributos tienen los mismos valores. Tráelo del laboratorio anterior si lo tienes implementado, pero haz que su cabecera sea exactamente la indicada.

```
@Override
public boolean equals(Object obj) {
   if (obj == null) return false;
   Email other = (Email) obj;
   return     …;
}
```

Parte complementaria

13) Añade los métodos implementados en la clase **`MailServer`** del laboratorio anterior. Analiza y plantea cómo podrías verificar el correcto comportamiento del método **`obtainAndRemoveEmail`**. No hagas uso del método **`getEmailList`** para su verificación.

14) Verifica el método **`obtainAndRemoveAllEmails.`**