

PRUEBAS UNITARIAS CON JUNIT

PRUEBAS

JUnit se trata de un Framework Open Source para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos Software. El framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.

Automatización de Pruebas

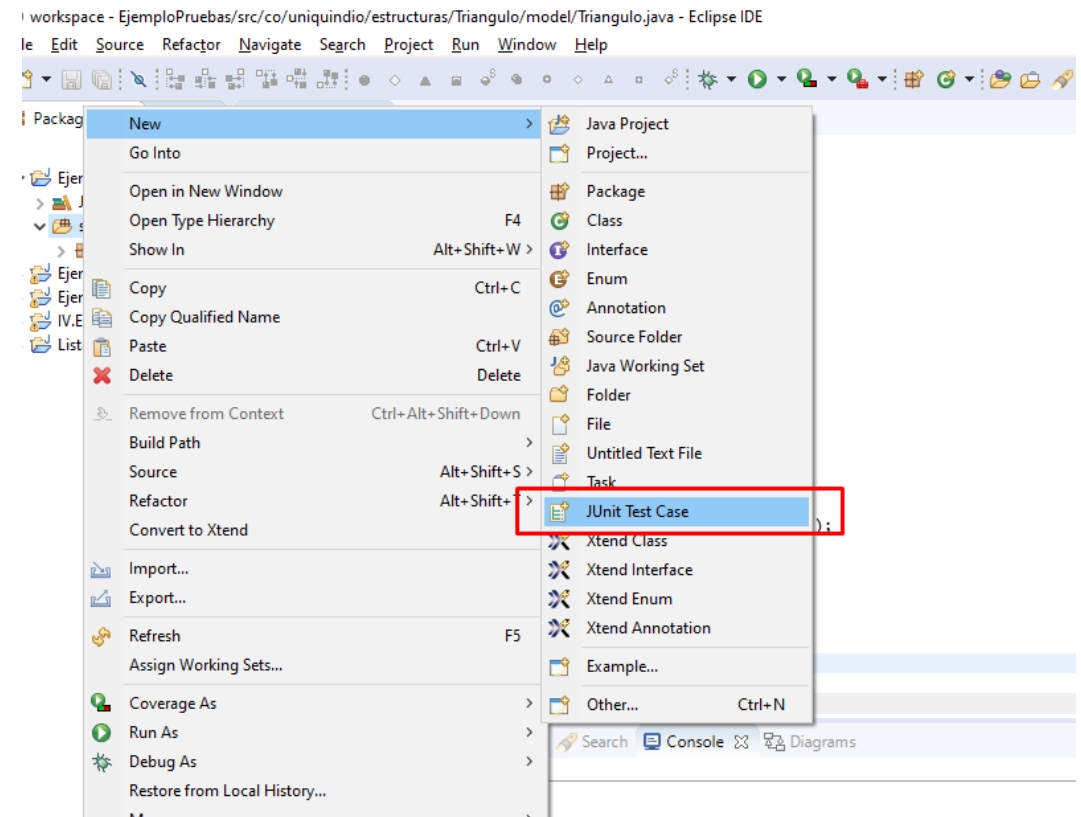
- ✗ No es muy apropiado llamar a una función, guardar el resultado en algún sitio y después tener que comprobar manualmente si el resultado era el deseado.
- ✓ Mantener automatizado un conjunto amplio de tests permite reducir el tiempo que se tarda en depurar errores y en verificar la corrección del código.

JUnit

JUnit se trata de un Framework Open Source para la automatización de las pruebas (tanto unitarias, como de integración) en los proyectos Software. El framework provee al usuario de herramientas, clases y métodos que le facilitan la tarea de realizar pruebas en su sistema y así asegurar su consistencia y funcionalidad.

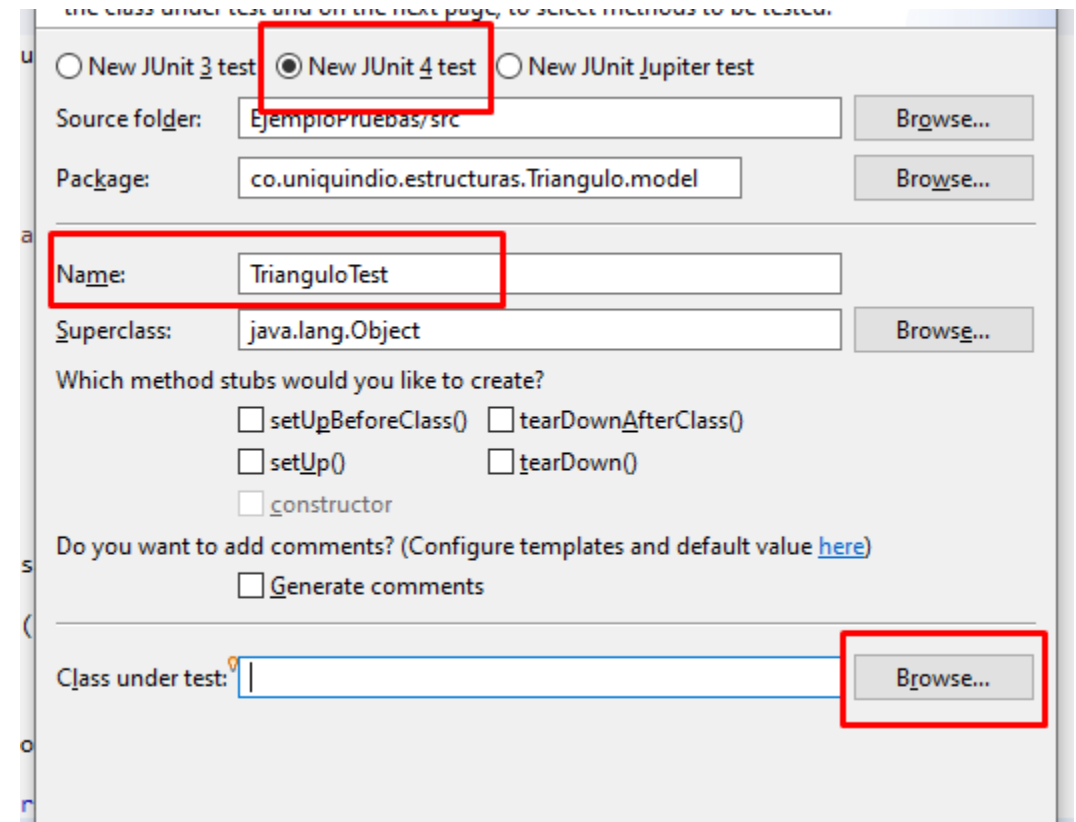
Como crearlas en eclipse

- Creamos un nueva clase para pruebas



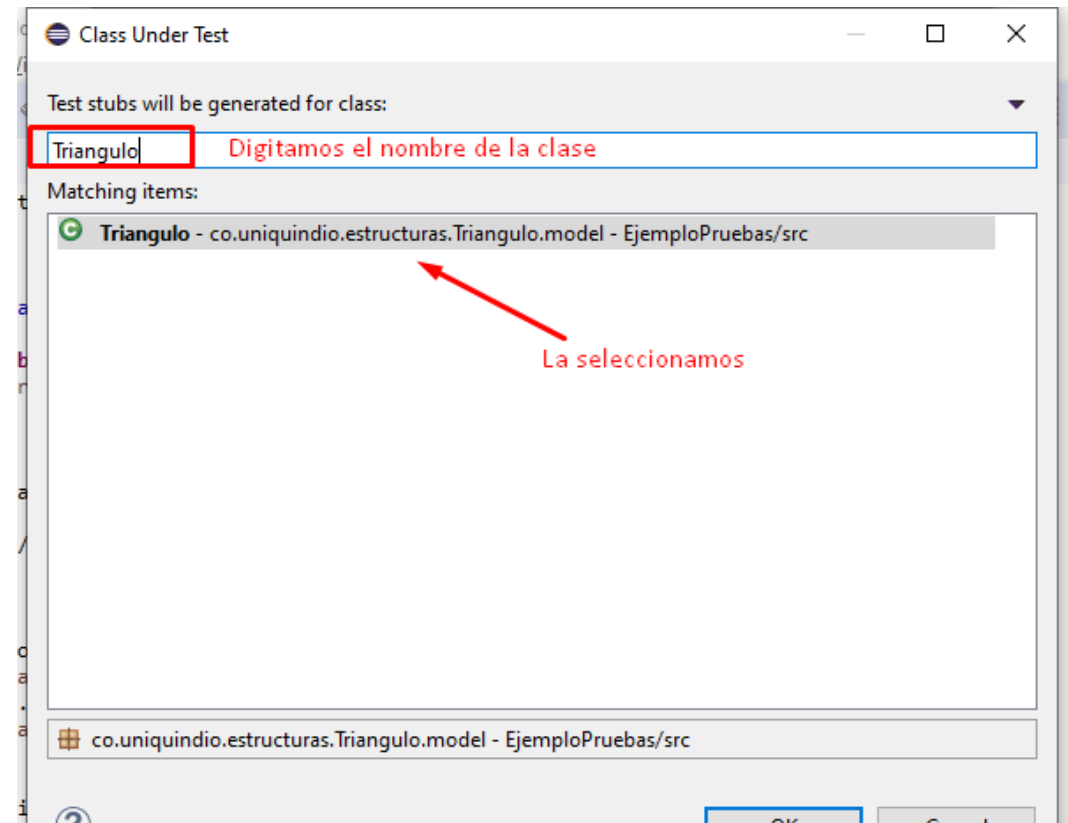
Como crearlas en eclipse

- Creamos un nueva clase para pruebas.
- Seleccionamos la version de Junit (v4)
- Nombre de la Clase
- Y buscamos la clase bajo prueba



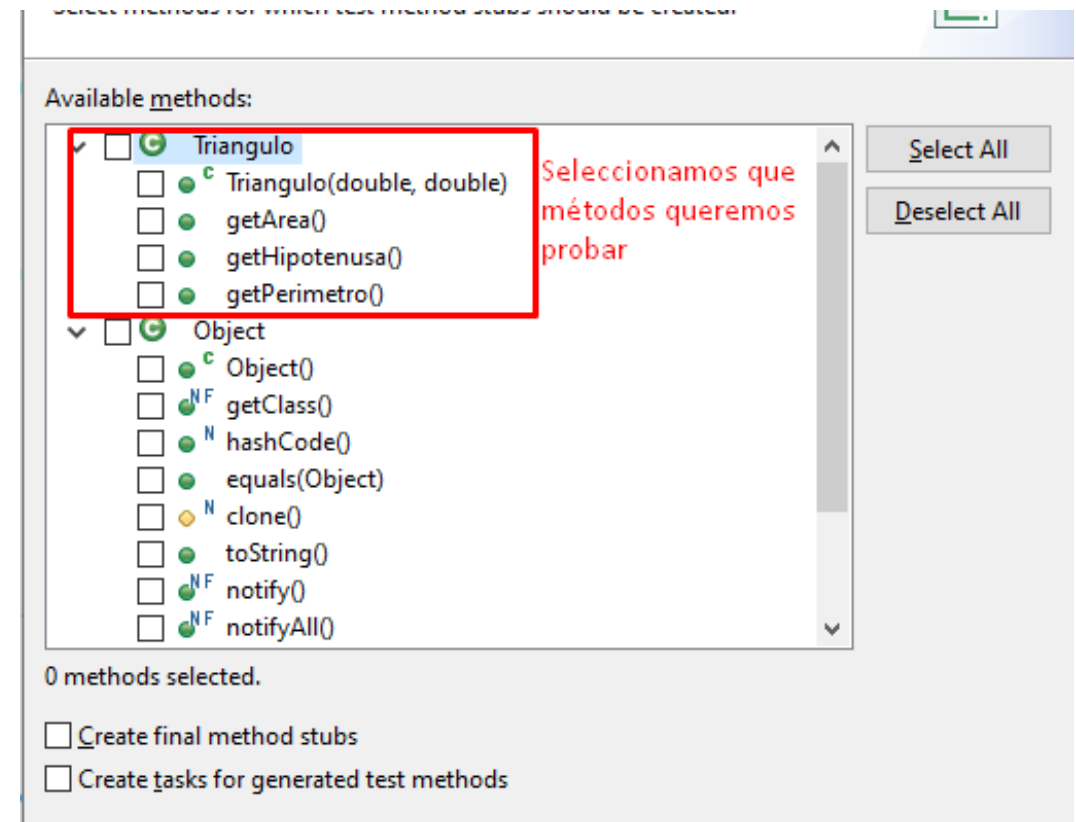
Como crearlas en eclipse

- Buscamos y seleccionamos la clase a la cual le queremos realizar pruebas



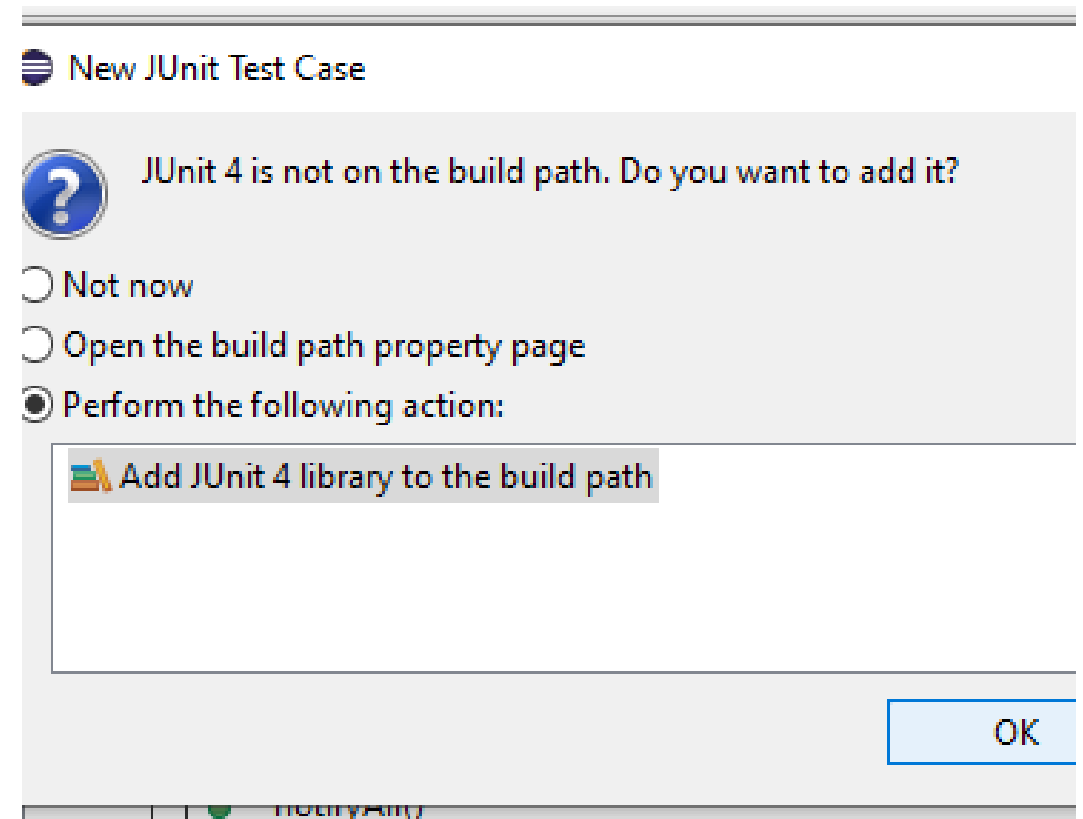
Como crearlas en eclipse

- Seleccionamos los metodos que queremos someter a pruebas



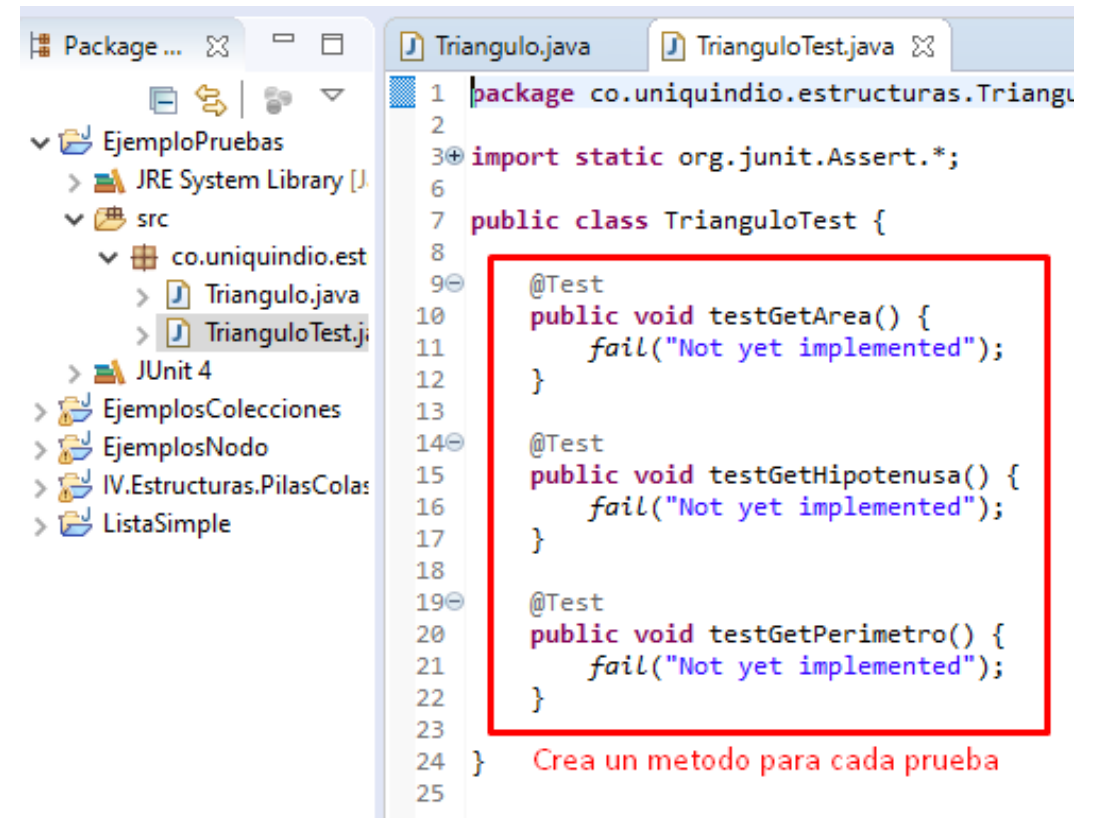
Como crearlas en eclipse

- Adicionamos la librería Junit al path del proyecto



Como crearlas en eclipse

- Crea un método por cada método que previamente seleccionado



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays a project structure with a package named `co.uniquindio.est` containing `Triangulo.java` and `TrianguloTest.java`. The `TrianguloTest.java` file is selected. The main editor shows the code for `TrianguloTest.java`, which includes package declarations, imports, and three test methods: `testGetArea()`, `testGetHipotenusa()`, and `testGetPerimetro()`. Each test method is annotated with `@Test` and contains a `fail("Not yet implemented");` statement. A red rectangle highlights these three test methods. Below the code, a red text annotation reads "Crea un metodo para cada prueba".

```
1 package co.uniquindio.estructuras.Triangu
2
3 import static org.junit.Assert.*;
6
7 public class TrianguloTest {
8
9     @Test
10     public void testGetArea() {
11         fail("Not yet implemented");
12     }
13
14     @Test
15     public void testGetHipotenusa() {
16         fail("Not yet implemented");
17     }
18
19     @Test
20     public void testGetPerimetro() {
21         fail("Not yet implemented");
22     }
23
24 }
25
```

Crea un metodo para cada prueba

Como crearlas en eclipse

- La anotación **@test** indica que el método es para ejecutar una prueba

```
package co.uniquindio.estructuras.Triangul

import static org.junit.Assert.*;

public class TrianguloTest {

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

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

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

}
```

Como crearlas en eclipse

- Se crea automáticamente el metodo fail el cual lanza un mensaje de fallo en la prueba

```
import static org.junit.Assert.*;

public class TrianguloTest {

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

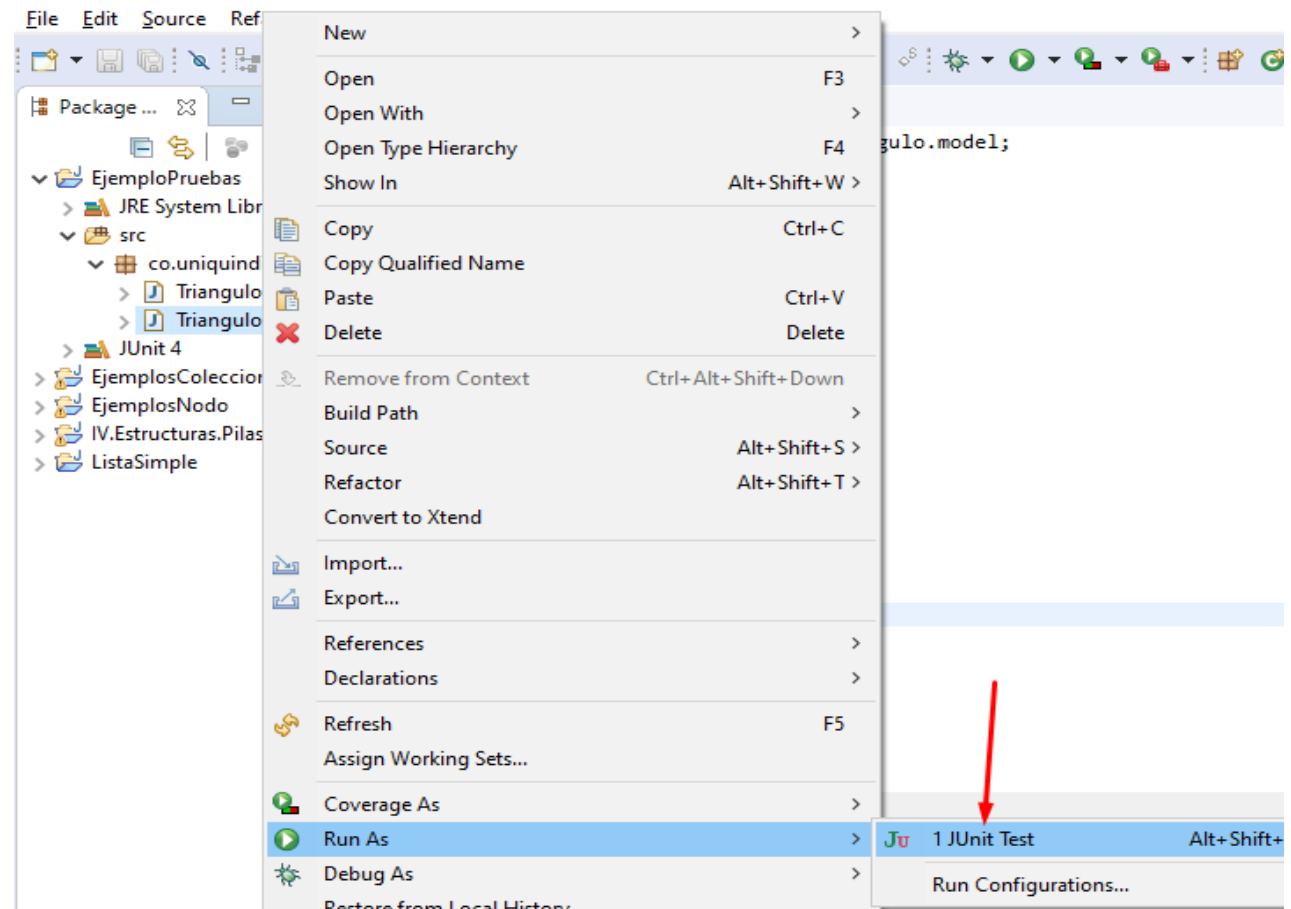
    @Test
    public void testGetHipotenusa() {
        fail("Not yet implemented");
    }

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

}
```

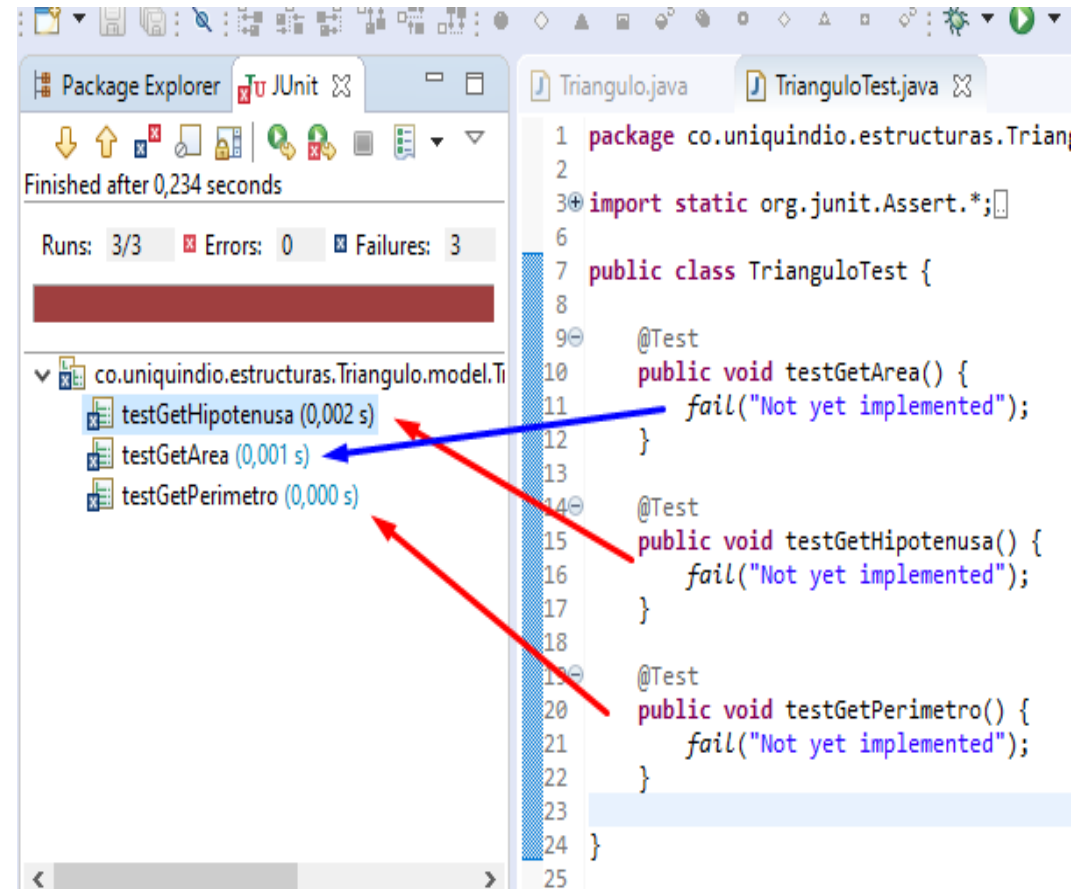
Como ejecutarlas

Ejecutamos la clase TrianguloTest como un **JUnit Test**



Al ejecutarlas

- Aparece el panel de JUnit, en el cual por cada metodos @test nos mostrará el resultado de la prueba, en este caso todos son Failures (fallos) ya que los metodos invocan a fail (lanzar falla)



Al ejecutarlas

- assertEquals verifica que la respuesta expect y la actual sea igual, con una tolerancia de delta (último parametro)

