
Entornos de Desarrollo

PRUEBAS UNITARIAS::JUNIT 5

DEFINICIÓN

Consiste en la implementación de los casos de prueba de caja blanca y caja negra identificados previamente.

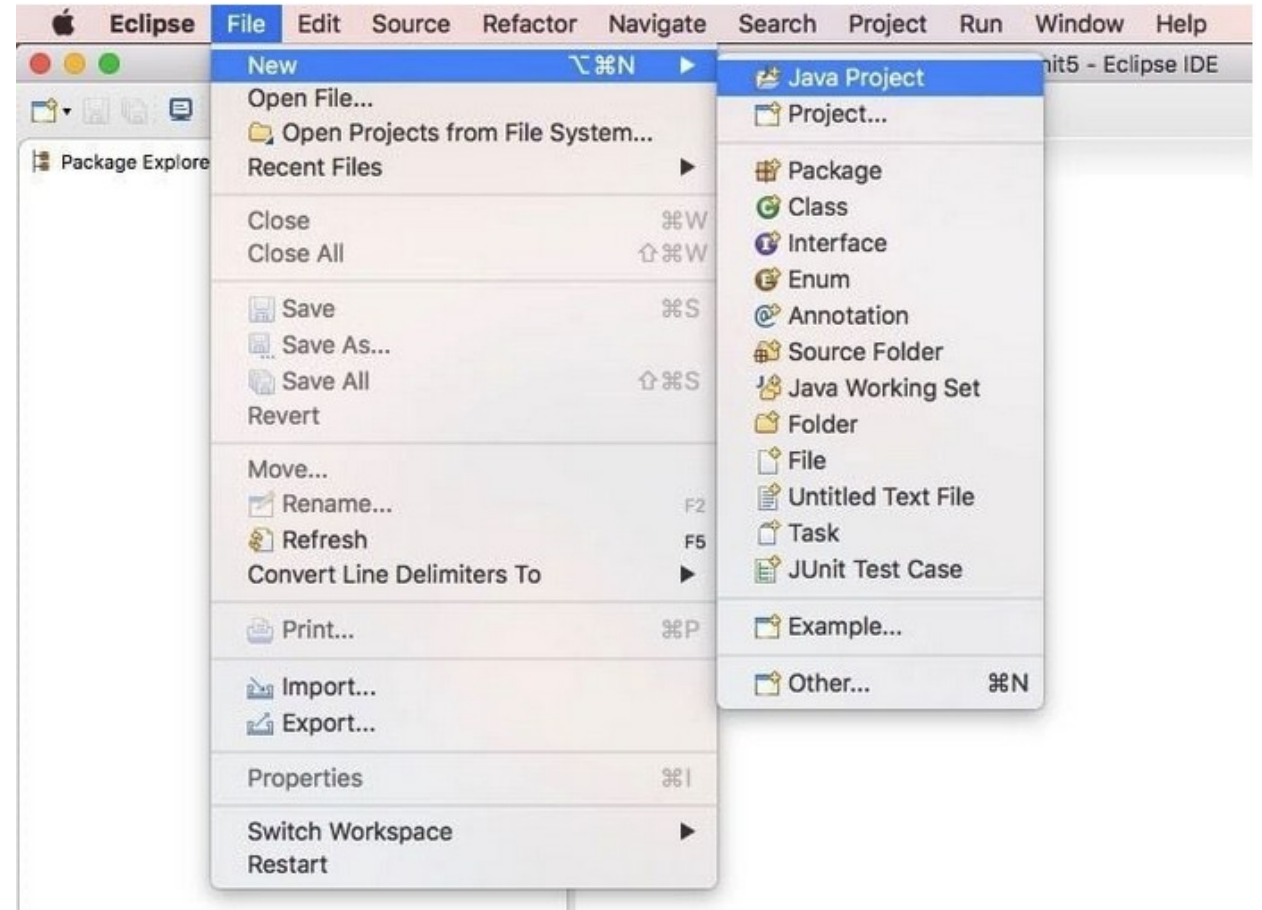
Para ello, disponemos de herramientas como **JUnit** y **Mockito**, comúnmente utilizadas en Java.

Para trabajar con cualquiera de dichas herramientas, deberemos descargar la librería **.jar** correspondiente o crear un proyecto con **Maven** donde se incluya como dependencia.

Trabajando con **Eclipse** también tenemos la posibilidad de incluir **JUnit** como un **plugin**.

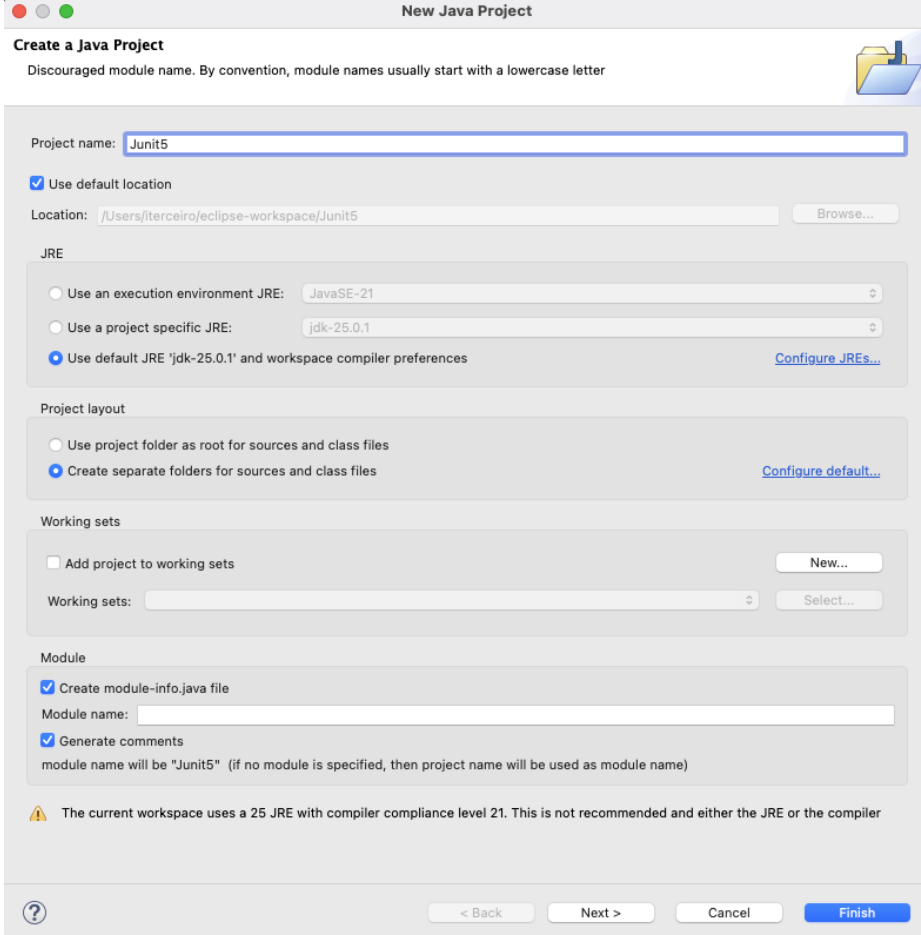
CREACIÓN DE UN PROYECTO JUNIT 5

Paso 1 - Ejecuta Eclipse IDE y crea un nuevo proyecto Java



CREACIÓN DE UN PROYECTO JUNIT 5

Paso 2 – Asigna un nombre al proyecto, por ejemplo Junit5 y selecciona un JDK superior a Java 8.



The screenshot shows the 'New Java Project' dialog box in the Eclipse IDE. The dialog is titled 'New Java Project' and has a subtitle 'Create a Java Project'. A note at the top right says 'Discouraged module name. By convention, module names usually start with a lowercase letter'. The 'Project name' field is filled with 'Junit5'. The 'Use default location' checkbox is checked. The 'Location' field shows the path '/Users/iterceiro/eclipse-workspace/Junit5'. Under the 'JRE' section, the 'Use default JRE 'jdk-25.0.1' and workspace compiler preferences' option is selected. The 'Project layout' section has 'Create separate folders for sources and class files' selected. In the 'Working sets' section, the 'Add project to working sets' checkbox is unchecked. The 'Module' section has 'Create module-info.java file' and 'Generate comments' checked. A warning icon and message at the bottom state: 'The current workspace uses a 25 JRE with compiler compliance level 21. This is not recommended and either the JRE or the compiler'. Navigation buttons at the bottom include '< Back', 'Next >', 'Cancel', and 'Finish'.

New Java Project

Create a Java Project

Discouraged module name. By convention, module names usually start with a lowercase letter

Project name: Junit5

☒ Use default location

Location: /Users/iterceiro/eclipse-workspace/Junit5 [Browse...](#)

JRE

☐ Use an execution environment JRE: JavaSE-21

☐ Use a project specific JRE: jdk-25.0.1

☒ Use default JRE 'jdk-25.0.1' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

☒ Create module-info.java file

Module name:

☒ Generate comments

module name will be "Junit5" (if no module is specified, then project name will be used as module name)

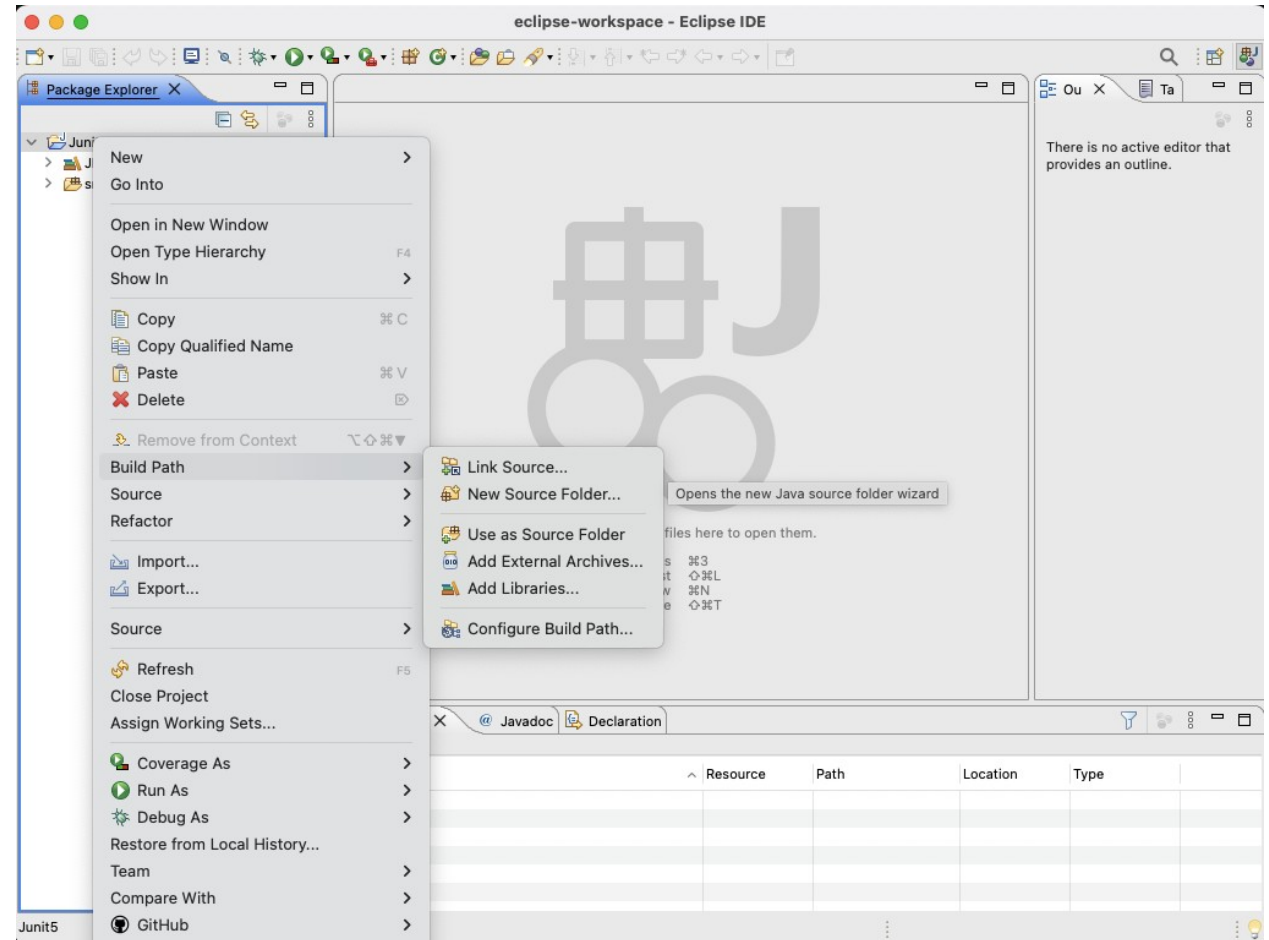
The current workspace uses a 25 JRE with compiler compliance level 21. This is not recommended and either the JRE or the compiler

[?](#) [< Back](#) [Next >](#) [Cancel](#) [Finish](#)

CREACIÓN DE UN PROYECTO JUNIT 5

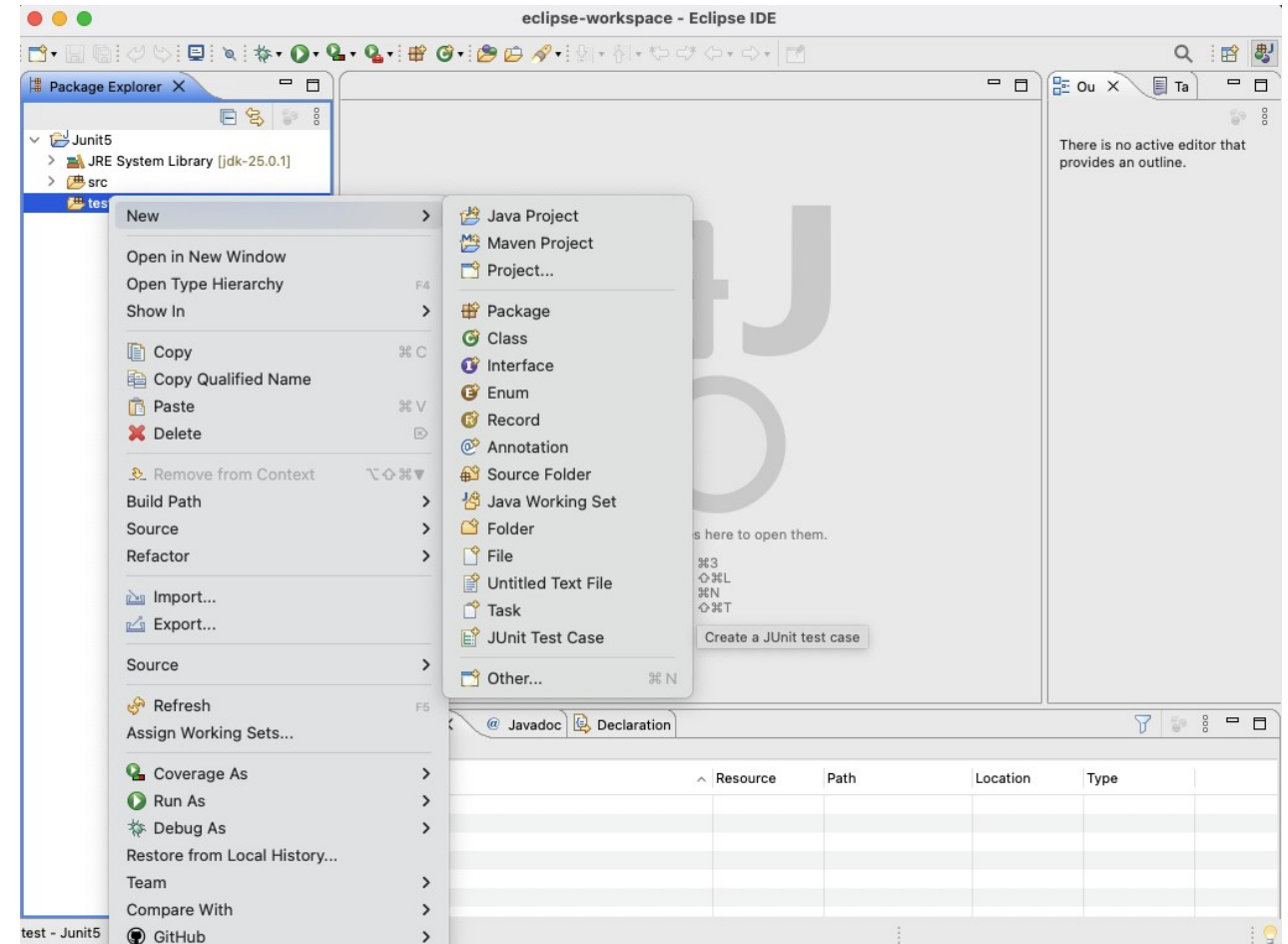
Paso 3 - El código fuente del programa se almacena en la carpeta **src**, pero los test de prueba se almacenan en la carpeta **test**.

Dicha carpeta tendremos que crearla manualmente, en el caso en que el IDE no la haya creado automáticamente al crear el proyecto.



CREACIÓN DE UN PROYECTO JUNIT 5

Paso 4 – Para implementar un caso de prueba, tenemos que crear una clase **JUnit Test Case**.

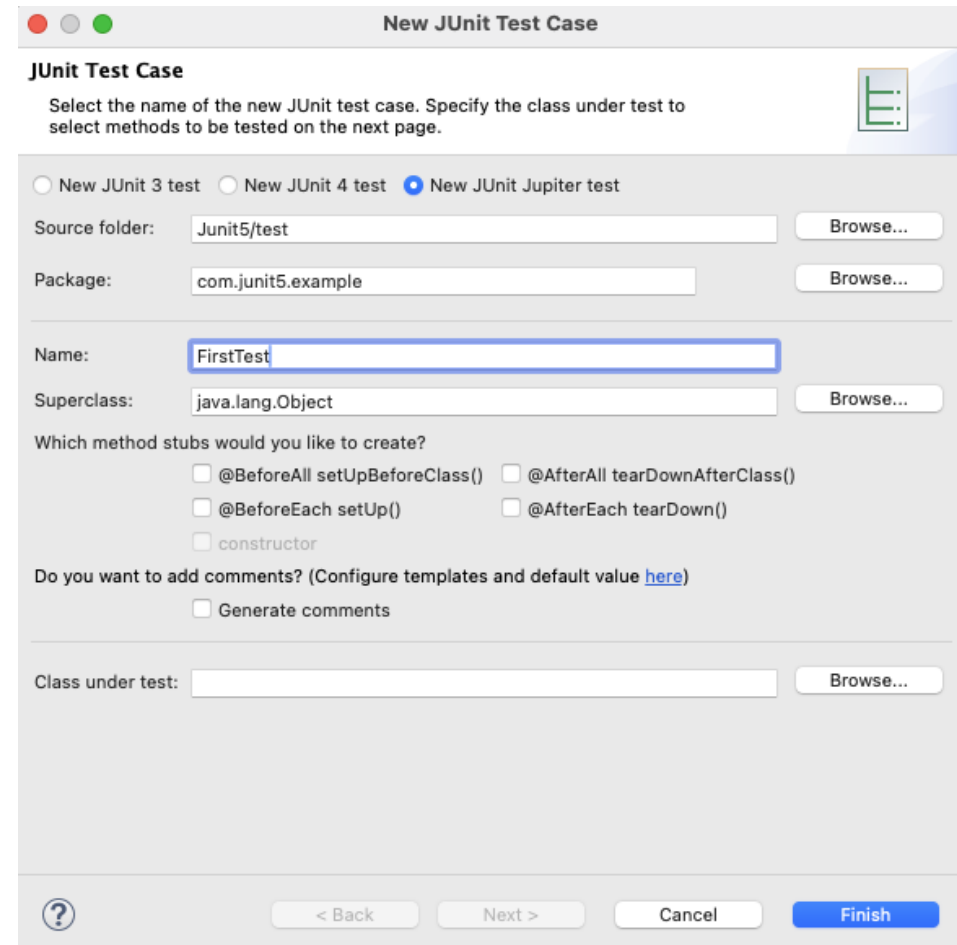


CREACIÓN DE UN PROYECTO JUNIT 5

Paso 5 – En el dialogo de creación de la clase de test, indica un nombre para la clase y un paquete.

Siempre es obligatorio especificar un nombre de paquete, que habitualmente es el dominio de la compañía, seguido del nombre del proyecto.

Por ejemplo, com.google.example



New JUnit Test Case

Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit Jupiter test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

☐ @BeforeAll setUpBeforeClass() ☐ @AfterAll tearDownAfterClass()
☐ @BeforeEach setUp() ☐ @AfterEach tearDown()
☐ constructor

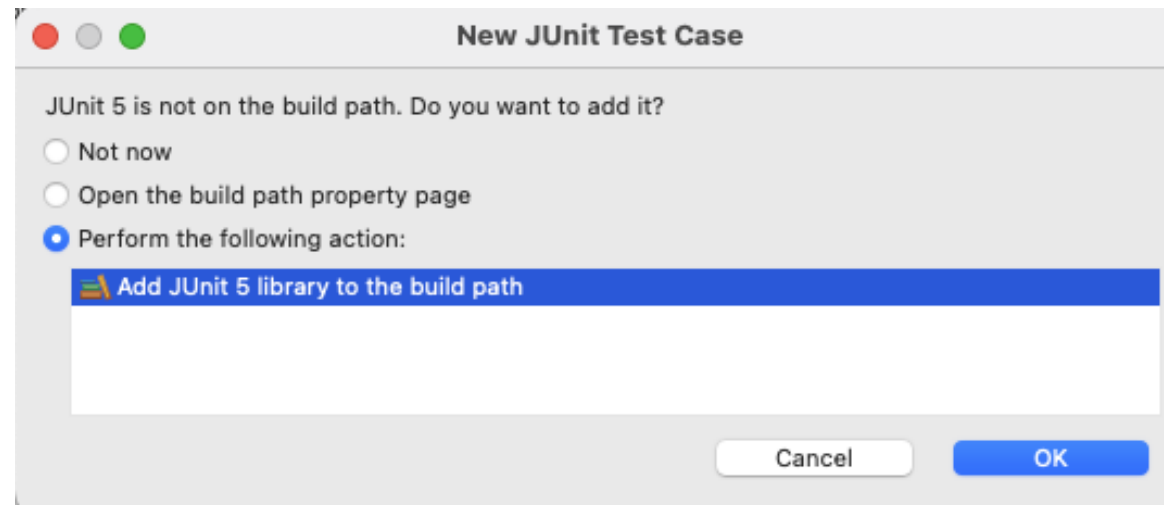
Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Class under test:

CREACIÓN DE UN PROYECTO JUNIT 5

Paso 6 - Al pulsar finalizar en el dialogo anterior, eclipse detecta que no tenemos instalada la librería o .jar de JUnit5 y nos pregunta si deseamos que se ocupe de descargarla e instalarla en nuestro proyecto o si deseamos realizarlo manualmente.

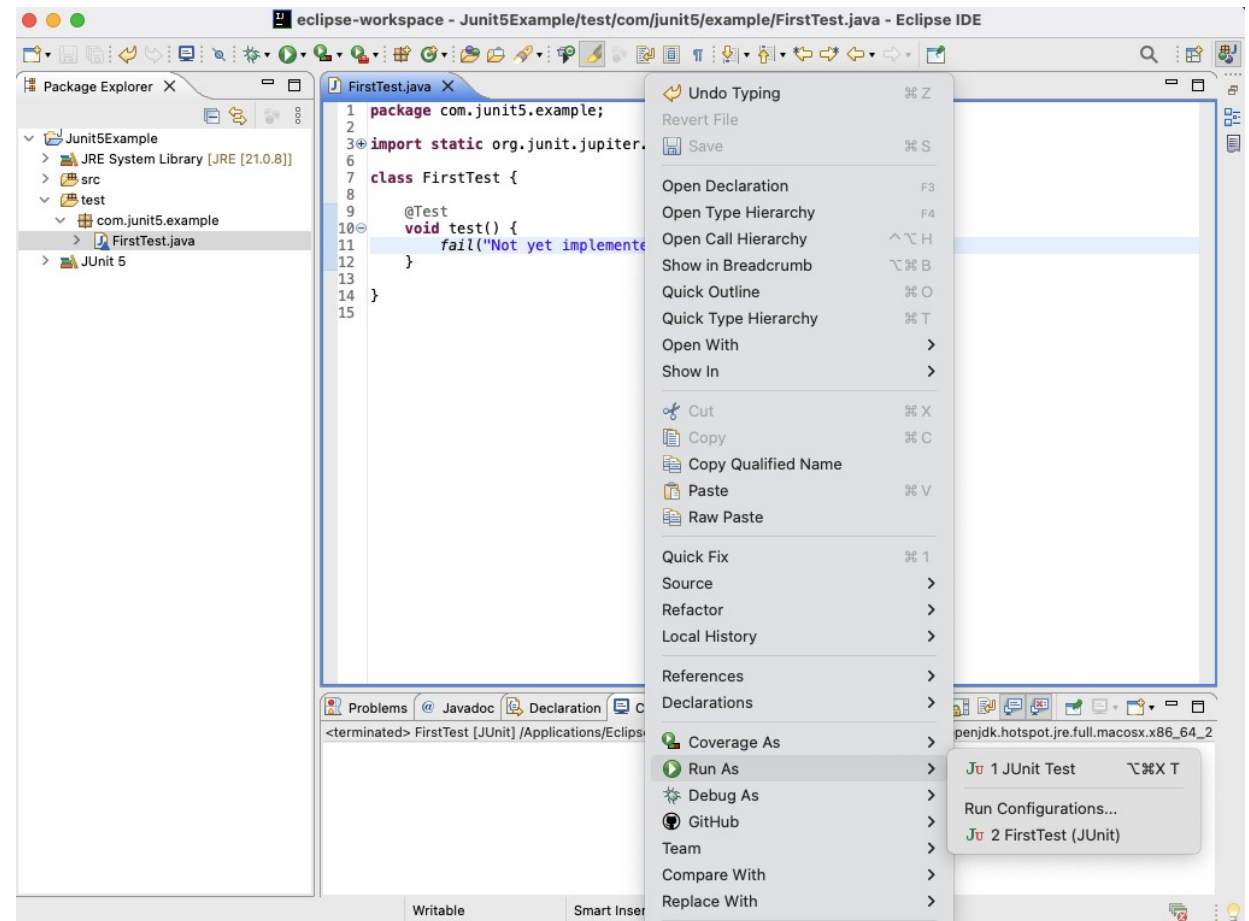
En este momento, aceptaremos la opción que nos propone para que realice la instalación de manera automática.



CREACIÓN DE UN PROYECTO JUNIT 5

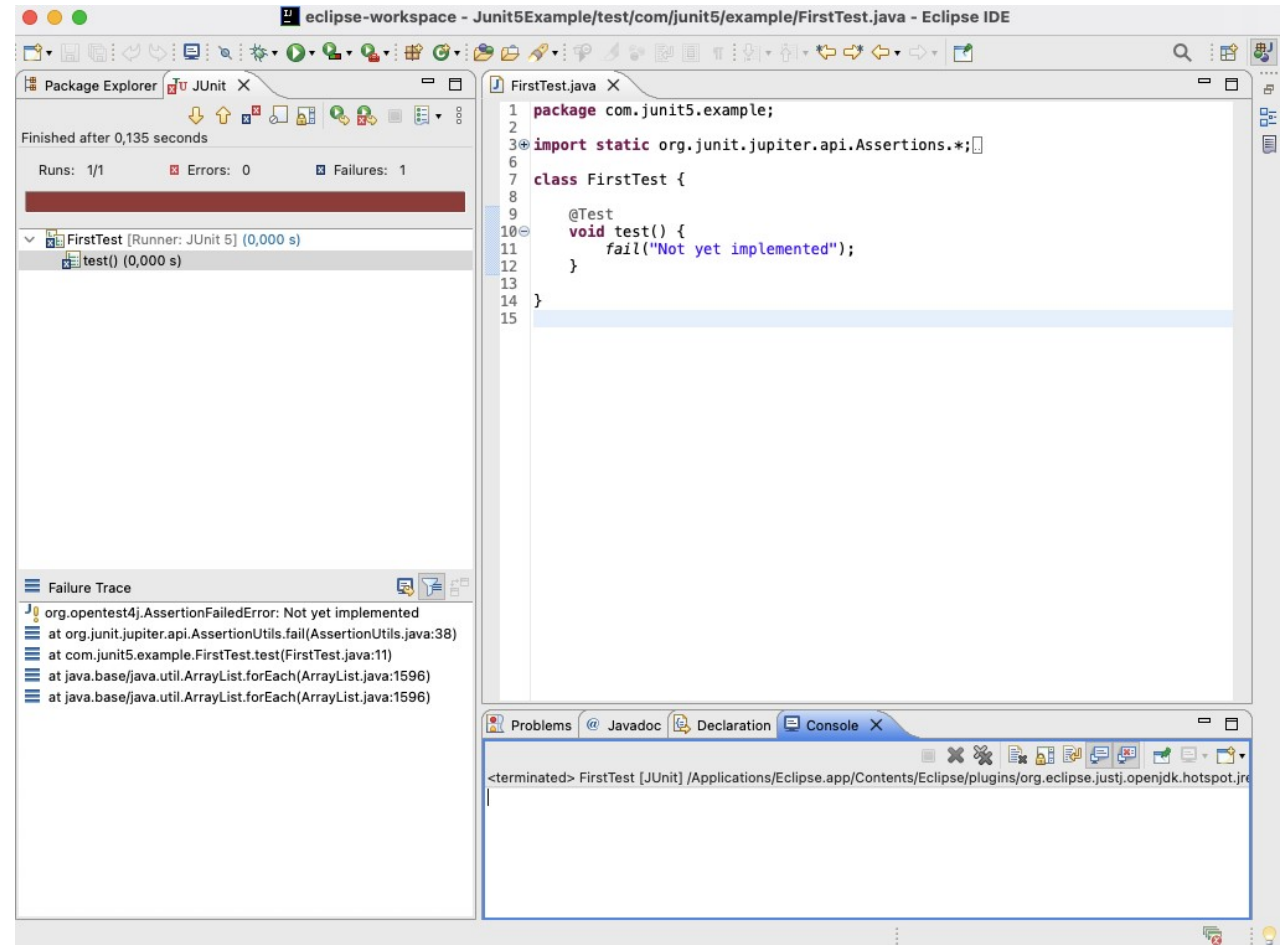
Paso 7 - Se habrá creado el código por defecto de un test JUnit.

Para ejecutarlo debemos seleccionar la opción **Run As → JUnit Test**.



CREACIÓN DE UN PROYECTO JUNIT 5

Paso 8 – El test fallará pues en el código tenemos implementado una **assertion** indicando que el test aún no ha sido implementado.



COMPONENTES JUNIT5

El framework **JUnit5** se divide en tres componentes principales:

- **Annotations:** conjunto de etiquetas o decoradores que permiten configurar los test.
- **Assertions:** métodos para verificar si el resultado de un test es el esperado.
- **Assumptions:** condiciones que si no se cumplen, el test es omitido en lugar de fallar.

ANNOTATIONS JUNIT5

Se trata de etiquetas que se anteponen a las declaraciones de código (clases, métodos, variables) para proporcionar información o modificar su comportamiento. Siempre empiezan con el símbolo @.

- **@Test**: marca un método como un caso de prueba a ejecutar.
- **@RepeatedTest**: el método será ejecutado el número de veces que se configure.
- **@BeforeEach**: el método se debe ejecutar antes de cada prueba.
- **@AfterEach**: el método se debe ejecutar después de cada prueba.
- **@Disabled**: Desactiva una prueba.

ANNOTATIONS JUNIT5

- **@BeforeAll**: el método se debe ejecutar una sola vez y antes que cualquier prueba.
- **@AfterAll**: el método se debe ejecutar una sola vez y después de todas las pruebas.
- **@ParametrizedTest**: el método va a recibir los valores de entrada configurados.
- **@ValueSource**: define el valor de cada parámetro del método.
- **@CsvSource**: define el conjunto de valores de los parámetros del método, para lanzar ejecuciones con varios conjuntos de entrada.
- **@DisplayName**: asigna un nombre personalizado al caso de prueba.

ASSERTIONS JUNIT5

Se trata de métodos que realizan una verificación de una condición. Si la condición no se cumple entonces la prueba es fallida. Por tanto, se utilizan para validar que el comportamiento de una sección de código es el esperado.

- **`assertEquals(expected, actual)`**: verifica que dos valores son iguales.
- **`assertTrue(condition)` y `assertFalse(condition)`**: verifica si una condición es verdadera o si es falsa.
- **`assertThrows(expectedException.class)`**: verifica que la excepción sea lanzada.
- **`assertAll(lambda → {...})`**: agrupa varias assertions para que se reporten todas juntas después de ejecutarse.

ASSUMPTIONS JUNIT5

Similares a las assertions, pero se utilizan cuando se desea NO marcar la prueba como fallida, sino que la aborta o salta.

Útil cuando no tiene sentido ejecutar una prueba bajo ciertas condiciones, como una variable de entorno o una propiedad del sistema.

- **assumeTrue(condition)**: aborta la prueba si la condición es falsa.
- **assumeFalse(condition)**: aborta la prueba si la condición es cierta.
- **assumingThat(condition, executable)**: ejecuta el bloque de código 'executable' cuando la condición es verdadera.