

# JUnit 5

---

JUnit es un framework Java para implementar test en Java. A diferencia de versiones anteriores, JUnit 5 se compone de tres sub-proyectos:

- **JUnit Platform.** Sirve como base sobre la cual se pueden ejecutar diferentes frameworks de pruebas. Es la plataforma que inicia el motor de pruebas y ejecuta las pruebas. A su vez se compone de:
  - **Launcher API:** permite a los clientes (IDEs, herramientas de construcción, etcétera...) descubrir y ejecutar pruebas.
  - **Test Engine API:** permite a los desarrolladores crear sus propios motores de pruebas que se pueden integrar con la plataforma. Por ejemplo se puede utilizar para ejecutar pruebas escritas en Spock, Cucumber, FitNesse, etcétera..., siempre y cuando haya un motor de pruebas compatible.
  - **Console Launcher:** una herramienta de línea de comandos para ejecutar pruebas.
- **JUnit Jupiter:** es la combinación del nuevo modelo de programación y la implementación del motor de pruebas para JUnit 5.
- **JUnit Vintage:** proporciona compatibilidad con versiones anteriores, permitiendo que las pruebas escritas con JUnit 3 y JUnit 4 se ejecuten en JUnit 5. Requiere que JUnit 4.12 esté presente en el class path o el module path.

JUnit 5 requiere **Java 8 (o superior)**.

## Anotaciones

---

JUnit se basa en [anotaciones](#):

- `@Test` : indica que el método que la contiene es un test.
- `@ParameterizedTest` : indica que ese método es un [test parametrizado](#).
- `@RepeatedTest` : indica que ese método es una plantilla para un [test repetible](#).
- `@Before` (JUnit4) / `@BeforeEach` (JUnit5): indica que ese método anotado debe ser ejecutado **antes** que cada método anotado como `@Test`, `RepeatedTest`, `@ParameterizedTest` o `@TestFactory`.
- `@After` (JUnit4) / `@AfterEach` (JUnit5): indica que ese método anotado debe ser ejecutado **después** que cada método anotado como `@Test`, `RepeatedTest`, `@ParameterizedTest` o `@TestFactory`.
- `@BeforeClass` (JUnit4) / `@BeforeAll` (JUnit5): indica que ese método anotado debe ser ejecutado **antes** que cualquier otro método anotado como `@Test`, `RepeatedTest`, `@ParameterizedTest` o `@TestFactory`.
- `@AfterClass` (JUnit4) / `@AfterAll` (JUnit5): indica que ese método anotado debe ser ejecutado **después** de todos los métodos anotado como `@Test`, `RepeatedTest`, `@ParameterizedTest` o `@TestFactory`.
- `@Ignore` (JUnit 4) / `@Disabled` (JUnit5): se utiliza para [deshabilitar el test](#).
- `@Tag` : se utiliza para declarar [etiquetas](#) que permitan filtrar por tests.
- `@Timeout` : se utiliza para fallar un test, una factoría de test, una plantilla de test o ciclo de vida si su ejecución excede una duración determinada.
- `@ExtendWith` : se utiliza para [registrar](#) extensiones de forma declarativa.

- `@DisplayName("cadena")` (JUnit5): Declara un nombre de visualización personalizado para la clase de prueba o el método de prueba. En lugar de usar esta anotación es recomendable utilizar nombres para los métodos lo suficientemente descriptivos como para que no sea necesario usar esta anotación.

Una clase estándar de prueba:

```
import static org.junit.jupiter.api.Assertions.fail;
import static org.junit.jupiter.api.Assumptions.assumeTrue;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.Test;

class StandardTests {

    @BeforeAll
    static void initAll() {
    }

    @BeforeEach
    void init() {
    }

    @Test
    void succeedingTest() {
    }

    @Test
    void failingTest() {
        fail("a failing test");
    }

    @Test
    @Disabled("for demonstration purposes")
    void skippedTest() {
        // not executed
    }

    @Test
    void abortedTest() {
        assumeTrue("abc".contains("Z"));
        fail("test should have been aborted");
    }

    @AfterEach
    void tearDown() {
    }

    @AfterAll
    static void tearDownAll() {
    }
}
```

## Aserciones

Las condiciones de aceptación del test se implementa con las [aserciones](#). Las más comunes son los siguientes:

- **assertTrue/assertFalse (condición a testear)**: Comprueba que la condición es cierta o falsa.
- **assertEquals/assertNotEquals (valor esperado, valor obtenido)**: Es importante el orden de los valores esperado y obtenido.

- **assertNull/assertNotNull (object)**: Comprueba que el objeto obtenido es nulo o no.
- **assertSame/assertNotSame(object1, object2)**: Comprueba si dos objetos son iguales o no.
- **fail()**: Fuerza que el test termine con fallo. Se puede indicar un mensaje.

```
class AssertionsTest {

    @Test
    void standardAssertions() {
        assertEquals(2, 2);
        assertEquals(4, 4, "Ahora el mensaje opcional de la aserción es el último parámetro.");
        assertTrue(2 == 2, () -> "Al usar una lambda para indicar el mensaje, "
            + "esta se evalúa cuando se va a mostrar (no cuando se ejecuta el assert), "
            + "de esta manera se evita el tiempo de construir mensajes complejos innecesariamente.");
    }

    @Test
    void groupedAssertions() {
        // En un grupo de aserciones se ejecutan todas ellas
        // y ser reportan todas los fallos juntos
        assertAll("user",
            () -> assertEquals("Francisco", user.getFirstName()),
            () -> assertEquals("Pérez", user.getLastName())
        );
    }

    @Test
    void exceptionTesting() {
        Throwable exception = expectThrows(IllegalArgumentException.class, () -> {
            throw new IllegalArgumentException("a message");
        });
        assertEquals("a message", exception.getMessage());
    }
}
```

## Tagging and Filtering

Las clases y métodos de prueba se pueden **etiquetar** via la anotación `@Tag`. Esto permite agrupar tests bajo ciertas etiquetas, facilitando la selección y ejecución de subconjuntos específicos de tests según sea necesario.

En JUnit 4, estas etiquetas se mapean a `@Category`.

```
import org.junit.jupiter.api.Tag;
import org.junit.jupiter.api.Test;

@Tag("fast")
@Tag("model")
class TaggingDemo {

    @Test
    @Tag("taxes")
    void testingTaxCalculation() {
    }

}
```

La sintaxis de las etiquetas debe seguir ciertas [reglas](#).

## Enlaces de interés

- [JUnit 5](#)
- <https://github.com/junit-team/junit5-samples>
- <https://www.baeldung.com/junit>
- <https://www.tutorialspoint.com/junit/index.htm>
- <https://www.vogella.com/tutorials/JUnit/article.html>

## Licencia

---



Esta obra está bajo una [licencia de Creative Commons Reconocimiento-Compartir Igual 4.0 Internacional](#).