

fail() method

This lesson demonstrates the importance of the fail method in JUnit 5 Assertions API.

WE'LL COVER THE FOLLOWING ^

- fail() method

fail() method

Assertions API provide static `fail()` method. As soon as, any `@Test` method encounters `fail()` static method, it will fail the test case. The primary usages of `fail()` method are as follows -

- It gives a piece of meaningful information to the programmer writing a test, that test case is in progress and still needs to be implemented.
- It can be used to verify that an actual exception is thrown. Usually based on some input when test case expects an exception at a certain line, providing `fail()` below that line will verify that exception was not thrown as code execution reached `fail()` method line. Thus, it explicitly fails the test case.

There are basically five useful overloaded methods to fail:-

```
public static void fail()
public static void fail(String message)
public static void fail(Supplier<String> messageSupplier)
public static void fail(Throwable throwable)
public static void fail(String message, Throwable throwable)
```



```
package io.educative.junit5;

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



```

import org.junit.jupiter.api.Test;

public class FailAssertionDemo {

    // usage 1 - @Test not implemented with fail()
    @Test
    public void testMethodYetNotImplemented() {
        fail();
    }

    // usage 2 - @Test not implemented with fail(String message)
    @Test
    public void testMethodYetNotImplemented1() {
        fail("@Test method not yet implemented !!!");
    }

    // usage 3 - @Test not implemented with fail(Supplier<String> messageSupplier)
    @Test
    public void testMethodYetNotImplemented2() {
        fail(() -> "@Test method not yet implemented !!!");
    }

    // usage 4 - @Test not implemented with fail(Throwable throwable)
    @Test
    public void testMethodYetNotImplemented3() {
        fail(new RuntimeException("@Test method not yet implemented !!!"));
    }

    // usage 5 - @Test not implemented with fail(String message, Throwable throwable)
    @Test
    public void testMethodYetNotImplemented4() {
        fail("@Test method not yet implemented !!!", new RuntimeException("Failed exp
    }

    // usage 6 - It can be used to verify that an actual exception is thrown
    @Test
    public void testActualExceptionThrown() {
        try {
            methodThatShouldThrowException();
            fail("Exception not thrown !!!");
        } catch (UnsupportedOperationException e) {
            // test case passed
        }
    }

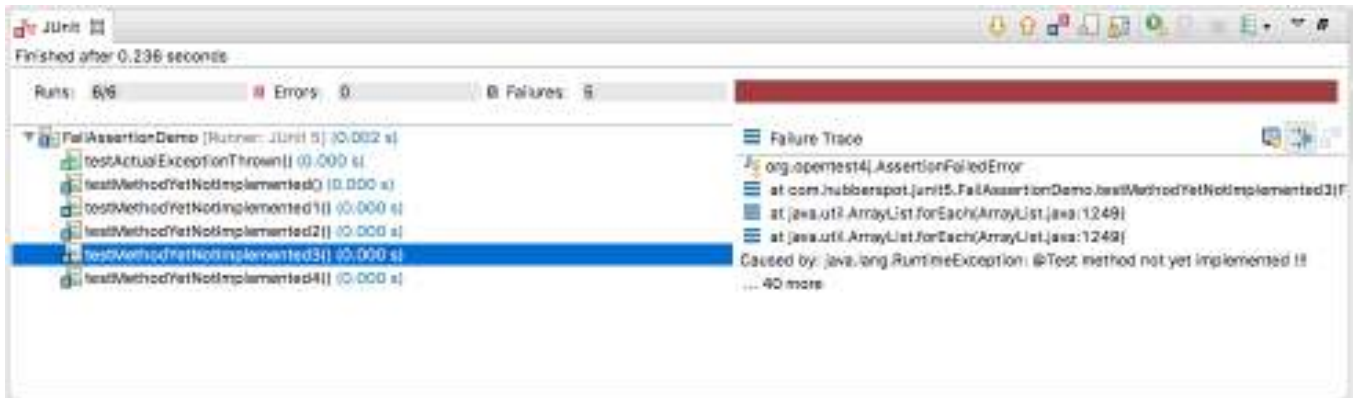
    private void methodThatShouldThrowException() {
        throw new UnsupportedOperationException(); // uncomment this line to will fail
    }
}

```



You can perform code changes to above code widget, run and practice different outcomes.

Step 4 - Run `FailedAssertionDemo.java` class as Junit Test.



In the next lesson, we will look into `assertTrue()` and `assertFalse()` assertion.