assertFalse method

This lesson demonstrates how to use assertFalse method in JUnit 5 to assert test conditions.

We'll cover the following

assertFalse() method

Assertions API provide static `assertFalse()` method. This method helps us in validating that the actual value supplied to it is `false`.

- If the actual value is `false` then test case will pass.
- If the actual value is `true` then test case will fail.

There are basically six useful overloaded methods for assertFalse -

```
1  public static void assertFalse(boolean condition)
2
3  public static void assertFalse(boolean condition, String message)
4
5  public static void assertFalse(boolean condition, Supplier<String> messageSupplier)
6
7  public static void assertFalse(BooleanSupplier booleanSupplier)
8
9  public static void assertFalse(BooleanSupplier booleanSupplier, String message)
10
11 public static void assertFalse(BooleanSupplier booleanSupplier, Supplier<String> message
```

assertFalse() method

Demo

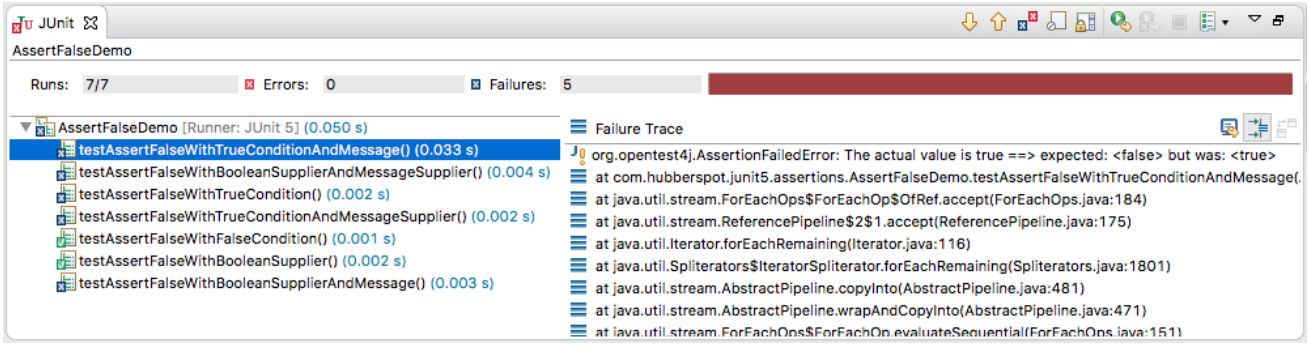
Let's look into the usage of the above methods.

```
package io.educative.junit5;

import static org.junit.jupiter.api.Assertions.assertFalse;

import org.junit.jupiter.api.Test;

public class AssertFalseDemo {

	@Test
	public void testAssertFalseWithFalseCondition() {
		boolean falseValue = false;
		assertFalse(falseValue);
	}

	@Test
	public void testAssertFalseWithTrueCondition() {
		boolean trueValue = true;
		assertFalse(trueValue);
	}

	@Test
	public void testAssertFalseWithTrueConditionAndMessage() {
		boolean trueValue = true;
		assertFalse(trueValue, "The actual value is true");
```

```
        }

        @Test
        public void testAssertFalseWithTrueConditionAndMessageSupplier() {
                boolean trueValue = true;
                assertFalse(trueValue, () -> "The actual value is true");
        }

        @Test
        public void testAssertFalseWithBooleanSupplier() {
                boolean falseValue = false;
                assertFalse(() -> falseValue);
        }

        @Test
        public void testAssertFalseWithBooleanSupplierAndMessage() {
                boolean trueValue = true;
                assertFalse(() -> trueValue, "The actual value is true");
        }

        @Test
        public void testAssertFalseWithBooleanSupplierAndMessageSupplier() {
                boolean trueValue = true;
                assertFalse(() -> trueValue, () -> "The actual value is true");
        }
}
```

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

Run AssertFalseDemo class as JUnit Test.



Explanation -

In `AssertFalseDemo` class, there are 7 `@Test` methods. These 7 methods demonstrate the working of the above 6 overloaded methods of `assertFalse`.

1. `testAssertFalseWithFalseCondition()` - It asserts the boolean value

1. `testAssertFalseWithFalseCondition()` - It asserts the boolean value provided to `assertFalse()` method. Here, the actual value passed to it is `false`. Thus, it passes the Junit test case because `assertFalse` asserts that value passed to it should be false.

2. `testAssertFalseWithTrueCondition()` - It asserts the boolean value provided to `assertFalse()` method. Here, actual value passed to it is `true`. Thus, it fails the Junit test case with `AssertionFailedError:` expected: <false> but was: <true> because value passed to `assertFalse` method is `true`.

3. `testAssertFalseWithTrueConditionAndMessage` - It asserts the boolean value provided to `assertFalse()` method. Here, actual value passed to it is `true`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is true ==> expected: <false> but was: <true> because value passed to `assertFalse` method is `true.` It gives `AssertionFailedError` followed `String message` we provide to assertFalse() method.

4. `testAssertFalseWithTrueConditionAndMessageSupplier` - It asserts the boolean value provided to `assertFalse()` method. Here, actual value passed to it is `true`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is true ==> expected: <false> but was: <true>** because value passed to `assertFalse` method is `true.` It gives `AssertionFailedError` followed by lazily evaluated `String message` we provide to assertFalse() method, as lambda expression.

5. `testAssertFalseWithBooleanSupplier()` - It asserts the boolean value provided to `assertFalse()` method through BooleanSupplier functional interface. Here, actual value passed to it is `false`. Thus, it passes the JUnit test case because `assertFalse` asserts that value passed to it should be false.

6. `testAssertFalseWithBooleanSupplierAndMessage` - It asserts the boolean value provided to `assertFalse()` method through BooleanSupplier functional interface. Here, actual value passed to it is `true`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is true ==> expected: <false> but was: <true> because value passed to `assertFalse` method is `true.` It gives `AssertionFailedError` followed `String message` we provide to assertFalse() method.

7. `testAssertFalseWithBooleanSupplierAndMessageSupplier` - It asserts the boolean value provided to `assertFalse()` method through BooleanSupplier functional interface. Here, actual value passed to it is `true`. Thus, it fails the Junit test case with `AssertionFailedError:` The actual value is true ==> expected: <false> but was: <true> because value passed to `assertFalse` method is `true.` It gives `AssertionFailedError` followed by lazily evaluated `String message` we provide to assertFalse() method, as lambda expression.

In the next lesson, we will look into `assertSame()` and `assertNotSame()` assertions.