JUnit (Java Unit) ist ein verbreitetes Framework, das sich zum Standard-werkzeug für automatisierte Unit-Tests von Klassen und Methoden in Java-Programmen etabliert hat. Ein JUnit-Test kennt nur zwei Ergebnisse: Entweder der Test gelingt oder er misslingt. Das Misslingen kann als Ursache einen Fehler (Error) oder ein falsches Ergebnis (Failure) haben, die beide per Exception signalisiert werden. Der Unterschied zwischen den beiden Begriffen liegt darin, dass Failures erwartet werden, während Errors eher unerwartet auftreten.

Quelle: „JUnit“ in Wikipedia URL: https://de.wikipedia.org/wiki/JUnit (Abgerufen: 25. Juli 2023)

Für den Testfall wird eine separate (Test-) Klasse erstellt, die mittels **Annotationen** in verschiedene Bereiche unterteilt werden kann. Nachfolgend eine (unvollständige) Zusammenstellung möglicher Methoden.

`@BeforeAll` `static void setUpBeforeClass() { ... }`	Kennzeichnung zur einmaligen Ausführung vor dem Aufruf aller anderen Testmethoden
`@AfterAll` `static void tearDownAfterClass(){ ... }`	Kennzeichnung zur einmaligen Ausführung nach dem Aufruf aller anderen Testmethoden
`@BeforeEach` `void setUp() { ... }`	Ausführung vor jedem Aufruf einer Testmethode zum Aufbau ein definierten Testumgebung
`@AfterEach` `void tearDown(){ ... }`	Ausführung nach jedem Aufruf einer Testmethode zur Erledigung von Aufräumarbeiten.
`@Test` `void test(){ ... }`	Kennzeichnung als Testfall (Testmethode)
`@DisplayName(" Beschriftung ")`	Ausgabe einer Beschriftung zur Testmethode
`@RepeatedTest(n)`	n-malige Wiederholung der Testmethode

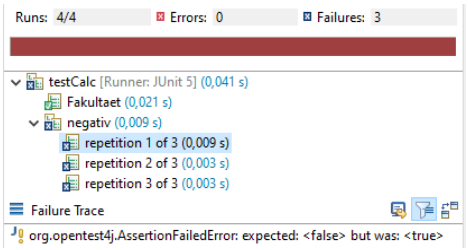
Die eigentlichen Testfälle werden mit **Assert-Methoden** durch-geführt. Diese werfen im Fehlerfall einen **java.lang.AssertionError**, der ggf. noch Details zum Fehler enthält. Zusätzlich kann mit dem letzten Parameter **String message**, der weitere Informationen zum Fehler enthalten kann, ausgegeben werden.

void assertTrue(condition)	Prüft, ob eine Bedingung wahr ist.
void assertFalse(condition)	Prüft, ob eine Bedingung falsch ist.
void assertNull(object)	Prüft, ob ein Objekt null ist.
void assertNull(object, String)	Wie assertNull() mit Fehlermessage.
void assertEquals(obj_1, obj_2)	Prüft ob zwei Objekte gleich sind.
void assertSame(obj_1, obj_2)	Prüft zwei Objektreferenzen.

Für die nebenstehende Klasse Calc mit der Methode **`int fak(int)`** ist ein einfaches Test-Szenario dargestellt.

```
①  import static org.junit.jupiter.api.Assertions.*;
   import org.junit.jupiter.api.*;
②  class testCalc {
     Calc C; int value;
③    @BeforeEach
     void setUp() {
       C = new Calc (); value = 5 ;  }
④    @Test
     @DisplayName("Fakultaet")
     void test1() {
       assertTrue(120 == Calc.fak (value));  }
⑤    @RepeatedTest(3)
     @DisplayName("negativ")
     void test2() {
       assertFalse(1 == Calc.fak (-1*value));}
⑥  }
```

```
public class Calc {
  public static long fak(int n) {
    int result = 1;
    for (int i = 2; i <= n; ++i) {
      result *= i;
    } return result; } }
```



Darstellung der Ausgabe in der IDE - eclipse