JUnit 5

Assertions all Method details

Method Detail

fail

public static <V> V fail(String message)

Fails a test with the given failure message.

See Javadoc for fail(String, Throwable) for an explanation of this method's generic return type V.

fail

public static <V> V fail(String message,
                         Throwable cause)

Fails a test with the given failure message as well as the underlying cause.

The generic return type V allows this method to be used directly as a single-statement lambda expression, thereby avoiding the need to implement a code block with an explicit return value. Since this method throws an AssertionFailedError before its return statement, this method never actually returns a value to its caller. The following example demonstrates how this may be used in practice.

```
Stream.of().map(entry -> fail("should not be called"));
```

fail

public static <V> V fail(Throwable cause)

Fails a test with the given underlying cause.

See Javadoc for fail(String, Throwable) for an explanation of this method's generic return type V.

fail

public static <V> V fail(Supplier<String> messageSupplier)

Fails a test with the failure message retrieved from the given messageSupplier.

See Javadoc for fail(String, Throwable) for an explanation of this method's generic return type V.

assertTrue

public static void assertTrue(boolean condition)

Asserts that the supplied condition is true.

assertTrue

public static void assertTrue(boolean condition,
                              Supplier<String> messageSupplier)

Asserts that the supplied condition is true.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertTrue

public static void assertTrue(BooleanSupplier booleanSupplier)

Asserts that the boolean condition supplied by booleanSupplier is true.

assertTrue

public static void assertTrue(BooleanSupplier booleanSupplier,
                              String message)

Asserts that the boolean condition supplied by booleanSupplier is true.

Fails with the supplied failure message.

assertTrue

public static void assertTrue(boolean condition,
                              String message)

Asserts that the supplied condition is true.

Fails with the supplied failure message.

assertTrue

public static void assertTrue(BooleanSupplier booleanSupplier,
                              Supplier<String> messageSupplier)

Asserts that the boolean condition supplied by booleanSupplier is true.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertFalse

public static void assertFalse(boolean condition)

Asserts that the supplied condition is not true.

assertFalse

public static void assertFalse(boolean condition,
String message)

Asserts that the supplied condition is not true.

Fails with the supplied failure message.

assertFalse

public static void assertFalse(boolean condition,
Supplier<String> messageSupplier)

Asserts that the supplied condition is not true.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertFalse

public static void assertFalse(BooleanSupplier booleanSupplier)

Asserts that the boolean condition supplied by booleanSupplier is not true.

assertFalse

public static void assertFalse(BooleanSupplier booleanSupplier,
String message)

Asserts that the boolean condition supplied by booleanSupplier is not true.

Fails with the supplied failure message.

assertFalse

public static void assertFalse(BooleanSupplier booleanSupplier,
Supplier<String> messageSupplier)

Asserts that the boolean condition supplied by booleanSupplier is not true.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertNull

```
public static void assertNull(Object actual)
```

Asserts that actual is null.

assertNull

```
public static void assertNull(Object actual,
                              String message)
```

Asserts that actual is null.

Fails with the supplied failure message.

assertNull

```
public static void assertNull(Object actual,
                              Supplier<String> messageSupplier)
```

Asserts that actual is null.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertNotNull

```
public static void assertNotNull(Object actual)
```

Asserts that actual is not null.

assertNotNull

```
public static void assertNotNull(Object actual,
                                 String message)
```

Asserts that actual is not null.

Fails with the supplied failure message.

assertNotNull

```
public static void assertNotNull(Object actual,
                                 Supplier<String> messageSupplier)
```

Asserts that actual is not null.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(short expected,
                                short actual)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(short expected,
                                short actual,
                                String message)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(short expected,
                                short actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(byte expected,
                                byte actual)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(byte expected,
                                byte actual,
                                String message)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(byte expected,
                                byte actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(int expected,
                                int actual)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(int expected,
                                int actual,
                                String message)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(int expected,
                                int actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(long expected,
                                long actual)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(long expected,
                                long actual,
                                String message)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(long expected,
                                long actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(char expected,
                                char actual)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(char expected,
                                char actual,
                                String message)
```

Asserts that expected and actual are equal.

assertEquals

```
public static void assertEquals(char expected,
                                char actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(float expected,
                                float actual)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertEquals

```
public static void assertEquals(float expected,
                                float actual,
                                String message)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertEquals

```
public static void assertEquals(float expected,
                                float actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(float expected,
                                float actual,
                                float delta)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertEquals

```
public static void assertEquals(float expected,
                                float actual,
                                float delta,
                                String message)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertEquals

```
public static void assertEquals(float expected,
                                float actual,
                                float delta,
                          Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(double expected,
                          double actual)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertEquals

```
public static void assertEquals(double expected,
                          double actual,
                          String message)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertEquals

```
public static void assertEquals(double expected,
                          double actual,
                          Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(double expected,
                                double actual,
                                double delta)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertEquals

```
public static void assertEquals(double expected,
                                double actual,
                                double delta,
                                String message)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertEquals

```
public static void assertEquals(double expected,
                                double actual,
                                double delta,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertEquals

```
public static void assertEquals(Object expected,
                                Object actual)
```

Asserts that expected and actual are equal.

If both are null, they are considered equal.

See Also:

Object.equals(Object)

assertEquals

```
public static void assertEquals(Object expected,
                                Object actual,
                                String message)
```

Asserts that expected and actual are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

See Also:

Object.equals(Object)

assertEquals

```
public static void assertEquals(Object expected,
                                Object actual,
                                Supplier<String> messageSupplier)
```

Asserts that expected and actual are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

Object.equals(Object)

assertArrayEquals

```
public static void assertArrayEquals(boolean[] expected,
                                     boolean[] actual)
```

Asserts that expected and actual boolean arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(boolean[] expected,
                                     boolean[] actual,
                                     String message)
```

Asserts that expected and actual boolean arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(boolean[] expected,
                                     boolean[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual boolean arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(char[] expected,
                                     char[] actual)
```

Asserts that expected and actual char arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(char[] expected,
                                     char[] actual,
                                     String message)
```

Asserts that expected and actual char arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(char[] expected,
                                     char[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual char arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(byte[] expected,
                                     byte[] actual)
```

Asserts that expected and actual byte arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(byte[] expected,
                                     byte[] actual,
                                     String message)
```

Asserts that expected and actual byte arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(byte[] expected,
                                     byte[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual byte arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(short[] expected,
                                     short[] actual)
```

Asserts that expected and actual short arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(short[] expected,
                                     short[] actual,
                                     String message)
```

Asserts that expected and actual short arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(short[] expected,
                                     short[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual short arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(int[] expected,
                                     int[] actual)
```

Asserts that expected and actual int arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(int[] expected,
                                     int[] actual,
```

String message)

Asserts that expected and actual int arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(int[] expected,
                                     int[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual int arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(long[] expected,
                                     long[] actual)
```

Asserts that expected and actual long arrays are equal.

If both are null, they are considered equal.

assertArrayEquals

```
public static void assertArrayEquals(long[] expected,
                                     long[] actual,
                                     String message)
```

Asserts that expected and actual long arrays are equal.

If both are null, they are considered equal.

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(long[] expected,
                                     long[] actual,
                        Supplier<String> messageSupplier)
```

Asserts that expected and actual long arrays are equal.

If both are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                        float[] actual)
```

Asserts that expected and actual float arrays are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                        float[] actual,
                        String message)
```

Asserts that expected and actual float arrays are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                        float[] actual,
                        Supplier<String> messageSupplier)
```

Asserts that expected and actual float arrays are equal.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                                     float[] actual,
                                     float delta)
```

Asserts that expected and actual float arrays are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                                     float[] actual,
                                     float delta,
                                     String message)
```

Asserts that expected and actual float arrays are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(float[] expected,
                                     float[] actual,
                                     float delta,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual float arrays are equal within the given delta.

Equality imposed by this method is consistent with Float.equals(Object) and Float.compare(float, float).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual)
```

Asserts that expected and actual double arrays are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual,
                                     String message)
```

Asserts that expected and actual double arrays are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual double arrays are equal.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual,
                                     double delta)
```

Asserts that expected and actual double arrays are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual,
                                     double delta,
                                     String message)
```

Asserts that expected and actual double arrays are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

Fails with the supplied failure message.

assertArrayEquals

```
public static void assertArrayEquals(double[] expected,
                                     double[] actual,
                                     double delta,
                                     Supplier<String> messageSupplier)
```

Asserts that expected and actual double arrays are equal within the given delta.

Equality imposed by this method is consistent with Double.equals(Object) and Double.compare(double, double).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertArrayEquals

```
public static void assertArrayEquals(Object[] expected,
                                     Object[] actual)
```

Asserts that expected and actual object arrays are deeply equal.

If both are null, they are considered equal.

Nested float arrays are checked as in assertEquals(float, float).

Nested double arrays are checked as in assertEquals(double, double).

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[])

assertArrayEquals

public static void assertArrayEquals(Object[] expected,
                                      Object[] actual,
                                      String message)

Asserts that expected and actual object arrays are deeply equal.

If both are null, they are considered equal.

Nested float arrays are checked as in assertEquals(float, float).

Nested double arrays are checked as in assertEquals(double, double).

Fails with the supplied failure message.

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[])

assertArrayEquals

public static void assertArrayEquals(Object[] expected,
                                      Object[] actual,
                                      Supplier<String> messageSupplier)

Asserts that expected and actual object arrays are deeply equal.

If both are null, they are considered equal.

Nested float arrays are checked as in assertEquals(float, float).

Nested double arrays are checked as in assertEquals(double, double).

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[])

assertIterableEquals

public static void assertIterableEquals(Iterable<?> expected,
                                         Iterable<?> actual)

Asserts that expected and actual iterables are deeply equal.

Similarly to the check for deep equality in assertArrayEquals(Object[], Object[]), if two iterables are encountered (including expected and actual) then their iterators must return equal elements in the same order as each other. **Note:** this means that the iterables *do not* need to be of the same type. Example:

```
import static java.util.Arrays.asList;
 . . .
Iterable<Integer> i0 = new ArrayList<>(asList(1, 2, 3));
Iterable<Integer> i1 = new LinkedList<>(asList(1, 2, 3));
assertIterableEquals(i0, i1); // Passes
```

If both expected and actual are null, they are considered equal.

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[]), assertArrayEquals(Object[], Object[])

assertIterableEquals

```
public static void assertIterableEquals(Iterable<?> expected,
                                        Iterable<?> actual,
                                        String message)
```

Asserts that expected and actual iterables are deeply equal.

Similarly to the check for deep equality in assertArrayEquals(Object[], Object[], String), if two iterables are encountered (including expected and actual) then their iterators must return equal elements in the same order as each other. **Note:** this means that the iterables *do not* need to be of the same type. Example:

```
import static java.util.Arrays.asList;
 . . .
Iterable<Integer> i0 = new ArrayList<>(asList(1, 2, 3));
Iterable<Integer> i1 = new LinkedList<>(asList(1, 2, 3));
```

```
assertIterableEquals(i0, i1); // Passes
```

If both expected and actual are null, they are considered equal.

Fails with the supplied failure message.

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[]), assertArrayEquals(Object[], Object[], String)

assertIterableEquals

```
public static void assertIterableEquals(Iterable<?> expected,
                                        Iterable<?> actual,
                                        Supplier<String> messageSupplier)
```

Asserts that expected and actual iterables are deeply equal.

Similarly to the check for deep equality in assertArrayEquals(Object[], Object[], Supplier), if two iterables are encountered (including expected and actual) then their iterators must return equal elements in the same order as each other. **Note:** this means that the iterables *do not* need to be of the same type. Example:

```
import static java.util.Arrays.asList;
 . . .
Iterable<Integer> i0 = new ArrayList<>(asList(1, 2, 3));
Iterable<Integer> i1 = new LinkedList<>(asList(1, 2, 3));
assertIterableEquals(i0, i1); // Passes
```

If both expected and actual are null, they are considered equal.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

Objects.equals(Object, Object), Arrays.deepEquals(Object[], Object[]), assertArrayEquals(Object[], Object[], Supplier)

assertLinesMatch

public static void assertLinesMatch(List<String> expectedLines,
                                    List<String> actualLines)

Asserts that expected list of Strings matches actual list.

This method differs from other assertions that effectively only check String.equals(Object), in that it uses the following staged matching algorithm:

For each pair of expected and actual lines do

1. check if expected.equals(actual) - if yes, continue with next pair
2. otherwise treat expected as a regular expression and check via String.matches(String) - if yes, continue with next pair
3. otherwise check if expected line is a fast-forward marker, if yes apply fast-forward actual lines accordingly (see below) and goto 1.

A valid fast-forward marker is an expected line that starts and ends with the literal >> and contains at least 4 characters. Examples:

>>>>
>> stacktrace >>
>> single line, non Integer.parse()-able comment >>
Skip arbitrary number of actual lines, until first matching subsequent expected line is found. Any character between the fast-forward literals are discarded.

">> 21 >>"
Skip strictly 21 lines. If they can't be skipped for any reason, an assertion error is raised.

Example showing all three kinds of expected line formats:

```
│  │  │    caught: AssertionFailedError: single line fail message
>> S T A C K T R A C E >>
│  │  │  duration: [\d]+ ms
│  │  │    status: ✗ FAILED
│  └─ test() finished after [\d]+ ms\.
```

```
  └─ JUnit Jupiter finished after [\d]+ ms\.
 Test plan execution finished. Number of all tests: 1


 Test run finished after [\d]+ ms
```

assertNotEquals

```
public static void assertNotEquals(Object unexpected,
                                   Object actual)
```

Asserts that expected and actual are not equal.

Fails if both are null.

See Also:
Object.equals(Object)

assertNotEquals

```
public static void assertNotEquals(Object unexpected,
                                   Object actual,
                                   String message)
```

Asserts that expected and actual are not equal.

Fails if both are null.

Fails with the supplied failure message.

See Also:
Object.equals(Object)

assertNotEquals

```
public static void assertNotEquals(Object unexpected,
                                   Object actual,
                                   Supplier<String> messageSupplier)
```

Asserts that expected and actual are not equal.

Fails if both are null.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

Object.equals(Object)

assertSame

public static void assertSame(Object expected,
                              Object actual)

Asserts that expected and actual refer to the same object.

assertSame

public static void assertSame(Object expected,
                              Object actual,
                              String message)

Asserts that expected and actual refer to the same object.

Fails with the supplied failure message.

assertSame

public static void assertSame(Object expected,
                              Object actual,
                              Supplier<String> messageSupplier)

Asserts that expected and actual refer to the same object.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertNotSame

public static void assertNotSame(Object unexpected,
                                 Object actual)

Asserts that expected and actual do not refer to the same object.

assertNotSame

public static void assertNotSame(Object unexpected,
                                 Object actual,
                                 String message)

Asserts that expected and actual do not refer to the same object.

Fails with the supplied failure message.

assertNotSame

public static void assertNotSame(Object unexpected,
                                Object actual,
                                Supplier<String> messageSupplier)

Asserts that expected and actual do not refer to the same object.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

assertAll

public static void assertAll(Executable... executables)
                throws MultipleFailuresError

Asserts that *all* supplied executables do not throw exceptions.

See Javadoc for assertAll(String, Stream) for an explanation of this method's exception handling semantics.

Throws:

MultipleFailuresError

See Also:

assertAll(String, Executable...), assertAll(Stream), assertAll(String, Stream)

assertAll

public static void assertAll(Stream<Executable> executables)
                throws MultipleFailuresError

Asserts that *all* supplied executables do not throw exceptions.

See Javadoc for assertAll(String, Stream) for an explanation of this method's exception handling semantics.

Throws:

MultipleFailuresError

See Also:

assertAll(Executable...), assertAll(String, Executable...), assertAll(String, Stream)

assertAll

public static void assertAll(String heading,

```
                    Executable... executables)
                throws MultipleFailuresError
```

Asserts that *all* supplied executables do not throw exceptions.

See Javadoc for assertAll(String, Stream) for an explanation of this method's exception handling semantics.

Throws:

MultipleFailuresError

See Also:

assertAll(Executable...), assertAll(Stream), assertAll(String, Stream)

assertAll

```
public static void assertAll(String heading,
                    Stream<Executable> executables)
                throws MultipleFailuresError
```

Asserts that *all* supplied executables do not throw exceptions.

If any supplied Executable throws an exception (i.e., a Throwable or any subclass thereof), all remaining executables will still be executed, and all exceptions will be aggregated and reported in a MultipleFailuresError. However, if an executable throws a *blacklisted* exception — for example, an OutOfMemoryError — execution will halt immediately, and the blacklisted exception will be rethrown *as is* but *masked* as an unchecked exception.

The supplied heading will be included in the message string for the MultipleFailuresError.

Throws:

MultipleFailuresError

See Also:

assertAll(Executable...), assertAll(String, Executable...), assertAll(Stream)

assertThrows

```
public static <T extends Throwable> T assertThrows(Class<T> expectedType,
                                    Executable executable)
```

Asserts that execution of the supplied executable throws an exception of the expectedType and returns the exception.

If no exception is thrown, or if an exception of a different type is thrown, this method will fail.

If you do not want to perform additional checks on the exception instance, simply ignore the return value.

assertThrows

```
public static <T extends Throwable> T assertThrows(Class<T> expectedType,
                                                   Executable executable,
                                                   String message)
```

Asserts that execution of the supplied executable throws an exception of the expectedType and returns the exception.

If no exception is thrown, or if an exception of a different type is thrown, this method will fail.

If you do not want to perform additional checks on the exception instance, simply ignore the return value.

assertThrows

```
public static <T extends Throwable> T assertThrows(Class<T> expectedType,
                                                   Executable executable,
                                                   Supplier<String> messageSupplier)
```

Asserts that execution of the supplied executable throws an exception of the expectedType and returns the exception.

If no exception is thrown, or if an exception of a different type is thrown, this method will fail.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

If you do not want to perform additional checks on the exception instance, simply ignore the return value.

assertTimeout

```
public static void assertTimeout(Duration timeout,
                                 Executable executable)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in the same thread as that of the calling code. Consequently, execution of the executable will not be preemptively aborted if the timeout is exceeded.

See Also:

assertTimeout(Duration, Executable, String), assertTimeout(Duration, Executable, Supplier), assertTimeout(Duration, ThrowingSupplier), assertTimeout(Duration, ThrowingSupplier, String), assertTimeout(Duration, ThrowingSupplier, Supplier), assertTimeoutPreemptively(Duration, Executable)

assertTimeout

```
public static void assertTimeout(Duration timeout,
                                 Executable executable,
                                 String message)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in the same thread as that of the calling code. Consequently, execution of the executable will not be preemptively aborted if the timeout is exceeded.

Fails with the supplied failure message.

See Also:

assertTimeout(Duration, Executable), assertTimeout(Duration, Executable, Supplier), assertTimeout(Duration, ThrowingSupplier), assertTimeout(Duration, ThrowingSupplier, String), assertTimeout(Duration, ThrowingSupplier, Supplier), assertTimeoutPreemptively(Duration, Executable, String)

assertTimeout

```
public static void assertTimeout(Duration timeout,
                                 Executable executable,
                                 Supplier<String> messageSupplier)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in the same thread as that of the calling code. Consequently, execution of the executable will not be preemptively aborted if the timeout is exceeded.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

assertTimeout(Duration, Executable), assertTimeout(Duration, Executable, String), assertTimeout(Duration, ThrowingSupplier), assertTimeout(Duration, ThrowingSupplier, String), assertTimeout(Duration, ThrowingSupplier, Supplier), assertTimeoutPreemptively(Duration, Executable, Supplier)

assertTimeout

public static <T> T assertTimeout(Duration timeout,
                                  ThrowingSupplier<T> supplier)

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in the same thread as that of the calling code. Consequently, execution of the supplier will not be preemptively aborted if the timeout is exceeded.

See Also:

assertTimeout(Duration, Executable), assertTimeout(Duration, Executable, String), assertTimeout(Duration, Executable, Supplier), assertTimeout(Duration, ThrowingSupplier, String), assertTimeout(Duration, ThrowingSupplier, Supplier), assertTimeoutPreemptively(Duration, Executable)

assertTimeout

public static <T> T assertTimeout(Duration timeout,
                                  ThrowingSupplier<T> supplier,
                                  String message)

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in the same thread as that of the calling code. Consequently, execution of the supplier will not be preemptively aborted if the timeout is exceeded.

Fails with the supplied failure message.

See Also:

assertTimeout(Duration, Executable), assertTimeout(Duration, Executable, String), assertTimeout(Duration, Executable, Supplier), assertTimeout(Duration, ThrowingSupplier), assertTimeout(Duration, ThrowingSupplier, Supplier), assertTimeoutPreemptively(Duration, Executable, String)

assertTimeout

```
public static <T> T assertTimeout(Duration timeout,
                                  ThrowingSupplier<T> supplier,
                                  Supplier<String> messageSupplier)
```

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in the same thread as that of the calling code. Consequently, execution of the supplier will not be preemptively aborted if the timeout is exceeded.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

assertTimeout(Duration, Executable), assertTimeout(Duration, Executable, String), assertTimeout(Duration, Executable, Supplier), assertTimeout(Duration, ThrowingSupplier), assertTimeout(Duration, ThrowingSupplier, String), assertTimeoutPreemptively(Duration, Executable, Supplier)

assertTimeoutPreemptively

```
public static void assertTimeoutPreemptively(Duration timeout,
                                             Executable executable)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in a different thread than that of the calling code. Furthermore, execution of the executable will be preemptively aborted if the timeout is exceeded.

See Also:

assertTimeoutPreemptively(Duration, Executable, String), assertTimeoutPreemptively(Duration, Executable, Supplier), assertTimeoutPreemptively(Duration, ThrowingSupplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, String), assertTimeoutPreemptively(Duration, ThrowingSupplier, Supplier), assertTimeout(Duration, Executable)

assertTimeoutPreemptively

```
public static void assertTimeoutPreemptively(Duration timeout,
                                             Executable executable,
                                             String message)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in a different thread than that of the calling code. Furthermore, execution of the executable will be preemptively aborted if the timeout is exceeded.

Fails with the supplied failure message.

See Also:

assertTimeoutPreemptively(Duration, Executable), assertTimeoutPreemptively(Duration, Executable, Supplier), assertTimeoutPreemptively(Duration, ThrowingSupplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, String), assertTimeoutPreemptively(Duration, ThrowingSupplier, Supplier), assertTimeout(Duration, Executable, String)

assertTimeoutPreemptively

```
public static void assertTimeoutPreemptively(Duration timeout,
                                             Executable executable,
                                             Supplier<String> messageSupplier)
```

Asserts that execution of the supplied executable completes before the given timeout is exceeded.

Note: the executable will be executed in a different thread than that of the calling code. Furthermore, execution of the executable will be preemptively aborted if the timeout is exceeded.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

assertTimeoutPreemptively(Duration, Executable), assertTimeoutPreemptively(Duration, Executable, String), assertTimeoutPreemptively(Duration, ThrowingSupplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, String), assertTimeoutPreemptively(Duration, ThrowingSupplier, Supplier), assertTimeout(Duration, Executable, Supplier)

assertTimeoutPreemptively

```
public static <T> T assertTimeoutPreemptively(Duration timeout,
                                             ThrowingSupplier<T> supplier)
```

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in a different thread than that of the calling code. Furthermore, execution of the supplier will be preemptively aborted if the timeout is exceeded.

See Also:

assertTimeoutPreemptively(Duration, Executable), assertTimeoutPreemptively(Duration, Executable, String), assertTimeoutPreemptively(Duration, Executable, Supplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, String), assertTimeoutPreemptively(Duration, ThrowingSupplier, Supplier), assertTimeout(Duration, Executable)

assertTimeoutPreemptively

```
public static <T> T assertTimeoutPreemptively(Duration timeout,
```

ThrowingSupplier<T> supplier,
String message)

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in a different thread than that of the calling code. Furthermore, execution of the supplier will be preemptively aborted if the timeout is exceeded.

Fails with the supplied failure message.

See Also:

assertTimeoutPreemptively(Duration, Executable), assertTimeoutPreemptively(Duration, Executable, String), assertTimeoutPreemptively(Duration, Executable, Supplier), assertTimeoutPreemptively(Duration, ThrowingSupplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, Supplier), assertTimeout(Duration, Executable, String)

assertTimeoutPreemptively

```
public static <T> T assertTimeoutPreemptively(Duration timeout,
                                              ThrowingSupplier<T> supplier,
                                              Supplier<String> messageSupplier)
```

Asserts that execution of the supplied supplier completes before the given timeout is exceeded.

If the assertion passes then the supplier's result is returned.

Note: the supplier will be executed in a different thread than that of the calling code. Furthermore, execution of the supplier will be preemptively aborted if the timeout is exceeded.

If necessary, the failure message will be retrieved lazily from the supplied messageSupplier.

See Also:

assertTimeoutPreemptively(Duration, Executable), assertTimeoutPreemptively(Duration, Executable, String), assertTimeoutPreemptively(Duration, Executable, Supplier), assertTimeoutPreemptively(Duration, ThrowingSupplier), assertTimeoutPreemptively(Duration, ThrowingSupplier, String), assertTimeout(Duration, Executable, Supplier)