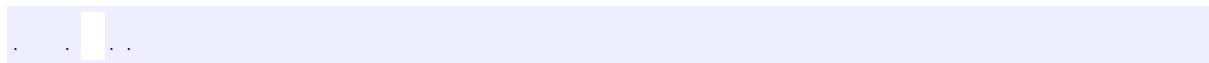


ÊùÓÐÀà [After](#)

[AfterClass](#)
[AllOf](#)
[AllTests](#)
[AnyOf](#)
[Assert](#)
[Assume](#)
[Before](#)
[BeforeClass](#)
[BlockJUnit4ClassRunner](#)
[ComparisonFailure](#)
[Computer](#)
[*Describable*](#)
[DescribedAs](#)
[Description](#)
[Failure](#)
[Filter](#)
[*Iterable*](#)
[Ignore](#)
[Is](#)
[IsAnything](#)
[isEqual](#)
[IsInstanceOf](#)
 [IsNot](#)
[IsNull](#)
[IsSame](#)
[JUnit4](#)
[JUnitCore](#)
[JUnitMatchers](#)
[NoTestsRemainException](#)
[Parameterized](#)
[Parameterized.Parameters](#)
[ParentRunner](#)
[Request](#)
[Result](#)
[Rule](#)
[RunListener](#)
[Runner](#)
[RunNotifier](#)
[RunWith](#)
[*Sortable*](#)
[Sorter](#)
[StoppedByUserException](#)
[Suite](#)
[Suite.SuiteClasses](#)
[Test](#)
[Test.None](#)

.....

.....



Deprecated API

- [Deprecated Methods](#)

Deprecated Methods

[org.junit.Assert.assertEquals\(double, double\)](#) *Use assertEquals(double
double actual, double epsilon) instead*

[org.junit.Assert.assertEquals\(Object\[\], Object\[\]\)](#)
use assertEquals

[org.junit.Assert.assertEquals\(String, double, double\)](#)
*Use assertEquals(String message, double expected, double actual,
epsilon) instead*

[org.junit.Assert.assertEquals\(String, Object\[\], Object\[\]\)](#)
use assertEquals

[org.junit.runner.Request.errorReport\(Class, Throwable\)](#)

[org.junit.runners.BlockJUnit4ClassRunner.possiblyExpectingExceptions\(Frame
Object, Statement\)](#)

Will be private soon: use Rules instead

[org.junit.runners.BlockJUnit4ClassRunner.validateInstanceMethods\(List\)](#)
unused API, will go away in future version

[org.junit.runners.BlockJUnit4ClassRunner.withAfters\(FrameworkMethod, Obj
Statement\)](#)

Will be private soon: use Rules instead

[org.junit.runners.BlockJUnit4ClassRunner.withBefores\(FrameworkMethod, Obj
Statement\)](#)

Will be private soon: use Rules instead

[org.junit.runners.BlockJUnit4ClassRunner.withPotentialTimeout\(FrameworkMe
Statement\)](#)

Will be private soon: use Rules instead





How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

The [Overview](#) page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (*italic*)
- Classes
- Enums
- Exceptions
- Errors
- Annotation Types

/

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
 - Direct Subclasses
 - All Known Subinterfaces
 - All Known Implementing Classes
 - Class/interface declaration
 - Class/interface description
-
- Nested Class Summary
 - Field Summary

- Constructor Summary
- Method Summary
- Field Detail
- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Annotation Type

Each annotation type has its own separate page with the following sections:

- Annotation Type declaration
- Annotation Type description
- Required Element Summary
- Optional Element Summary
- Element Detail

Enum

Each enum has its own separate page with the following sections:

- Enum declaration
- Enum description
- Enum Constant Summary
- Enum Constant Detail

O

There is a [Class Hierarchy](#) page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the

- hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

Deprecated API

The [Deprecated API](#) page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The [Index](#) contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take you to the next or previous class, interface, package, or related page.

Frames/No Frames

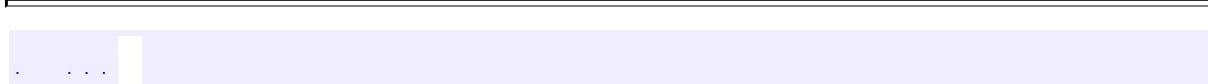
These links show and hide the HTML frames. All pages are available with or without frames.

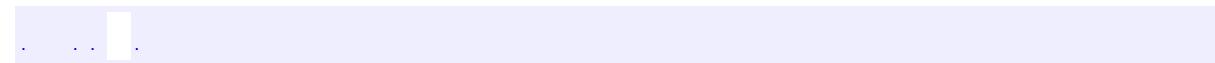
Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementors, not to developers using the API. While there is no link in the navigation bar, you can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

The [.page](#) lists the static final fields and their values.

This help file applies to API documentation generated using the standard doclet.





A B C D E F G H I J M N O P R S T V W

A

[**aClass\(Class<?>\)**](#) - Static method in class org.junit.runner.[Request](#)

Create a Request that, when processed, will run all the tests in a class.

[**addChild\(Description\)**](#) - Method in class org.junit.runner.[Description](#)

Add Description as a child of the receiver.

[**addFirstListener\(RunListener\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Internal use only.

[**addListener\(RunListener\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Add a listener to be notified as the tests run.

[**addListener\(RunListener\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Internal use only

[**After**](#) - Annotation Type in [org.junit](#)

If you allocate external resources in a [Before](#) method you need to release them after the test runs.

[**AfterClass**](#) - Annotation Type in [org.junit](#)

If you allocate expensive external resources in a [BeforeClass](#) method you need to release them after all the tests in the class have run.

[**ALL**](#) - Static variable in class org.junit.runner.manipulation.[Filter](#)

A null Filter that passes all tests through.

[**AllOf<T>**](#) - Class in [org.hamcrest.core](#)

Calculates the logical conjunction of two matchers.

[**AllOf\(Iterable<Matcher<? extends T>>\)**](#) - Constructor for class org.hamcrest.core.[AllOf](#)

[**allOf\(Matcher<? extends T>...\)**](#) - Static method in class

org.hamcrest.core.[AllOf](#)

Evaluates to true only if ALL of the passed in matchers evaluate to true.

[**allOf\(Iterable<Matcher<? extends T>>\)**](#) - Static method in class org.hamcrest.core.[AllOf](#)

Evaluates to true only if ALL of the passed in matchers evaluate to true.

[**AllTests**](#) - Class in [org.junit.runners](#)

Runner for use with JUnit 3.8.x-style AllTests classes (those that only implement a static suite() method).

[**AllTests\(Class<?>\)**](#) - Constructor for class org.junit.runners.[AllTests](#)

Only called reflectively.

[**any\(Class<T>\)**](#) - Static method in class org.hamcrest.core.[IsAnything](#)

This matcher always evaluates to true.

[**AnyOf<T>**](#) - Class in [org.hamcrest.core](#)

Calculates the logical disjunction of two matchers.

[**AnyOf\(Iterable<Matcher<? extends T>>\)**](#) - Constructor for class org.hamcrest.core.[AnyOf](#)

[**anyOf\(Matcher<? extends T>...\)**](#) - Static method in class

org.hamcrest.core.[AnyOf](#)

Evaluates to true if ANY of the passed in matchers evaluate to true.

[**anyOf\(Iterable<Matcher<? extends T>>\)**](#) - Static method in class org.hamcrest.core.[AnyOf](#)

Evaluates to true if ANY of the passed in matchers evaluate to true.

[**anything\(\)**](#) - Static method in class org.hamcrest.core.[IsAnything](#)

This matcher always evaluates to true.

[**anything\(String\)**](#) - Static method in class org.hamcrest.core.[IsAnything](#)

This matcher always evaluates to true.

[**apply\(Object\)**](#) - Method in class org.junit.runner.manipulation.[Filter](#)

Invoke with a [Runner](#) to cause all tests it intends to run to first be checked with the filter.

[**apply\(Object\)**](#) - Method in class org.junit.runner.manipulation.[Sorter](#)

Sorts the test in runner using comparator

[**Assert**](#) - Class in [org.junit](#)

A set of assertion methods useful for writing tests.

[**Assert\(\)**](#) - Constructor for class org.junit.[Assert](#)

Protect constructor since it is a static only class

[**assertArrayEquals\(String, Object\[\], Object\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two object arrays are equal.

[**assertArrayEquals\(Object\[\], Object\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two object arrays are equal.

[**assertArrayEquals\(String, byte\[\], byte\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two byte arrays are equal.

[**assertArrayEquals\(byte\[\], byte\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two byte arrays are equal.

[**assertArrayEquals\(String, char\[\], char\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two char arrays are equal.

[**assertArrayEquals\(char\[\], char\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two char arrays are equal.

[**assertArrayEquals\(String, short\[\], short\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two short arrays are equal.

[**assertArrayEquals\(short\[\], short\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two short arrays are equal.

[**assertArrayEquals\(String, int\[\], int\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two int arrays are equal.

[**assertArrayEquals\(int\[\], int\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two int arrays are equal.

[**assertArrayEquals\(String, long\[\], long\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two long arrays are equal.

[**assertArrayEquals\(long\[\], long\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two long arrays are equal.

[**assertArrayEquals\(String, double\[\], double\[\], double\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two double arrays are equal.

[**assertArrayEquals\(double\[\], double\[\], double\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two double arrays are equal.

[**assertArrayEquals\(String, float\[\], float\[\], float\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two float arrays are equal.

[**assertArrayEquals\(float\[\], float\[\], float\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two float arrays are equal.

[**assertEquals\(String, Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects are equal.

[**assertEquals\(Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects are equal.

[**assertEquals\(String, double, double, double\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two doubles or floats are equal to within a positive delta.

[**assertEquals\(long, long\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two longs are equal.

[**assertEquals\(String, long, long\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two longs are equal.

[**assertEquals\(double, double\)**](#) - Static method in class org.junit.[Assert](#)

Deprecated. Use `assertEquals(double expected, double actual, double epsilon)` instead

[**assertEquals\(String, double, double\)**](#) - Static method in class org.junit.[Assert](#)

Deprecated. Use `assertEquals(String message, double expected, double actual, double epsilon)` instead

[**assertEquals\(double, double, double\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two doubles or floats are equal to within a positive delta.

[**assertEquals\(String, Object\[\], Object\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Deprecated. use `assertArrayEquals`

[**assertEquals\(Object\[\], Object\[\]\)**](#) - Static method in class org.junit.[Assert](#)

Deprecated. use `assertArrayEquals`

[**assertFalse\(String, boolean\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that a condition is false.

[**assertFalse\(boolean\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that a condition is false.

[**assertNotNull\(String, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that an object isn't null.

[**assertNotNull\(Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that an object isn't null.

[**assertNotSame\(String, Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects do not refer to the same object.

[**assertNotSame\(Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects do not refer to the same object.

[**assertNull\(String, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that an object is null.

[**assertNull\(Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that an object is null.

[**assertSame\(String, Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects refer to the same object.

[**assertSame\(Object, Object\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that two objects refer to the same object.

[**assertThat\(T, Matcher<T>\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that actual satisfies the condition specified by matcher.

[**assertThat\(String, T, Matcher<T>\)**](#) - Static method in class org.junit.[Assert](#)

Asserts that actual satisfies the condition specified by matcher.

[assertTrue\(String, boolean\)](#) - Static method in class org.junit.[Assert](#)

Asserts that a condition is true.

[assertTrue\(boolean\)](#) - Static method in class org.junit.[Assert](#)

Asserts that a condition is true.

[Assume](#) - Class in [org.junit](#)

A set of methods useful for stating assumptions about the conditions in which a test is meaningful.

[Assume\(\)](#) - Constructor for class org.junit.[Assume](#)

[assumeNoException\(Throwable\)](#) - Static method in class org.junit.[Assume](#)

Use to assume that an operation completes normally.

[assumeNotNull\(Object...\)](#) - Static method in class org.junit.[Assume](#)

If called with one or more null elements in objects, the test will halt and be ignored.

[assumeThat\(T, Matcher<T>\)](#) - Static method in class org.junit.[Assume](#)

Call to assume that actual satisfies the condition specified by matcher.

[assumeTrue\(boolean\)](#) - Static method in class org.junit.[Assume](#)

If called with an expression evaluating to false, the test will halt and be ignored.

B

[Before](#) - Annotation Type in [org.junit](#)

When writing tests, it is common to find that several tests need similar objects created before they can run.

[BeforeClass](#) - Annotation Type in [org.junit](#)

Sometimes several tests need to share computationally expensive setup (like logging into a database).

[BlockJUnit4ClassRunner](#) - Class in [org.junit.runners](#)

Implements the JUnit 4 standard test case class model, as defined by the annotations in the org.junit package.

[BlockJUnit4ClassRunner\(Class<?>\)](#) - Constructor for class

[org.junit.runners.BlockJUnit4ClassRunner](#)

Creates a BlockJUnit4ClassRunner to run klass

[both\(Matcher<T>\)](#) - Static method in class [org.junit.matchers.JUnitMatchers](#)

This is useful for fluently combining matchers that must both pass.

C

[**childlessCopy\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**childrenInvoker\(RunNotifier\)**](#) - Method in class
org.junit.runners.[ParentRunner](#)

Returns a Statement: Call [ParentRunner.runChild\(Object, RunNotifier\)](#) on each object returned by [ParentRunner.getChildren\(\)](#) (subject to any imposed filter and sort)

[**classBlock\(RunNotifier\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

Constructs a Statement to run all of the tests in the test class.

[**classes\(Computer, Class<?>...\)**](#) - Static method in class org.junit.runner.[Request](#)

Create a Request that, when processed, will run all the tests in a set of classes.

[**classes\(Class<?>...\)**](#) - Static method in class org.junit.runner.[Request](#)

Create a Request that, when processed, will run all the tests in a set of classes with the default Computer.

[**classWithoutSuiteMethod\(Class<?>\)**](#) - Static method in class
org.junit.runner.[Request](#)

Create a Request that, when processed, will run all the tests in a class.

[**collectInitializationErrors\(List<Throwable>\)**](#) - Method in class
org.junit.runners.[BlockJUnit4ClassRunner](#)

[**collectInitializationErrors\(List<Throwable>\)**](#) - Method in class
org.junit.runners.[ParentRunner](#)

Adds to errors a throwable for each problem noted with the test class (available from [ParentRunner.getTestClass\(\)](#)).

[**compare\(Description, Description\)**](#) - Method in class
org.junit.runner.manipulation.[Sorter](#)

[**ComparisonFailure**](#) - Error in [org.junit](#)

Thrown when an [assertEquals\(String, String\)](#) fails.

[**ComparisonFailure\(String, String, String\)**](#) - Constructor for error
org.junit.[ComparisonFailure](#)

Constructs a comparison failure.

[**Computer**](#) - Class in [org.junit.runner](#)

Represents a strategy for computing runners and suites.

[**Computer\(\)**](#) - Constructor for class org.junit.runner.[Computer](#)

[**computeTestMethods\(\)**](#) - Method in class
org.junit.runners.[BlockJUnit4ClassRunner](#)

Returns the methods that run tests.

[**containsString\(String\)**](#) - Static method in class
org.junit.matchers.[JUnitMatchers](#)

[**createListener\(\)**](#) - Method in class org.junit.runner.[Result](#)

Internal use only.

[**createSuiteDescription\(String, Annotation...\)**](#) - Static method in class
org.junit.runner.[Description](#)

Create a Description named name.

[**createSuiteDescription\(Class<?>\)**](#) - Static method in class
org.junit.runner.[Description](#)

Create a Description named after testClass

[**createTest\(\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Returns a new fixture for running a test.

[**createTestDescription\(Class<?>, String, Annotation...\)**](#) - Static method in
class org.junit.runner.[Description](#)

Create a Description of a single test named name in the class clazz.

[**createTestDescription\(Class<?>, String\)**](#) - Static method in class
org.junit.runner.[Description](#)

Create a Description of a single test named name in the class clazz.

D

[Describable](#) - Interface in [org.junit.runner](#)

Represents an object that can describe itself

[describe\(\)](#) - Method in class org.junit.runner.manipulation.[Filter](#)

Returns a textual description of this Filter

[describeChild\(FrameworkMethod\)](#) - Method in class

org.junit.runners.[BlockJUnit4ClassRunner](#)

[describeChild\(T\)](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a [Description](#) for child, which can be assumed to be an element
of the list returned by [ParentRunner.getChildren\(\)](#)

[describeChild\(Runner\)](#) - Method in class org.junit.runners.[Suite](#)

[DescribedAs<T>](#) - Class in [org.hamcrest.core](#)

Provides a custom description to another matcher.

[DescribedAs\(String, Matcher<T>, Object\[\]\)](#) - Constructor for class

org.hamcrest.core.[DescribedAs](#)

[describedAs\(String, Matcher<T>, Object...\)](#) - Static method in class

org.hamcrest.core.[DescribedAs](#)

Wraps an existing matcher and overrides the description when it fails.

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[AllOf](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[AnyOf](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[DescribedAs](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[Is](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[IsAnything](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[isEqual](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[IsInstanceOf](#)

[describeTo\(Description\)](#) - Method in class org.hamcrest.core.[IsNot](#)

[**describeTo**\(**Description**\)](#) - Method in class org.hamcrest.core.[IsNull](#)

[**describeTo**\(**Description**\)](#) - Method in class org.hamcrest.core.[IsSame](#)

- Class in [org.junit.runner](#)

A **Description** describes a test which is to be run or has been run.

E

[**either\(Matcher<T>\)**](#) - Static method in class org.junit.matchers.[JUnitMatchers](#)

This is useful for fluently combining matchers where either may pass, for example:

[**EMPTY**](#) - Static variable in class org.junit.runner.[Description](#)

Describes a Runner which runs no tests

[**emptySuite\(\)**](#) - Static method in class org.junit.runners.[Suite](#)

Returns an empty suite.

[**equals\(Object\)**](#) - Method in class org.junit.runner.[Description](#)

[**equalTo\(T\)**](#) - Static method in class org.hamcrest.core.[IsEqual](#)

Is the value equal to another value, as tested by the

`Object.equals(java.lang.Object)` invokedMethod?

[**errorReport\(Class<?>, Throwable\)**](#) - Static method in class

org.junit.runner.[Request](#)

Deprecated.

[**everyItem\(Matcher<T>\)**](#) - Static method in class

org.junit.matchers.[JUnitMatchers](#)

F

[**fail\(String\)**](#) - Static method in class org.junit.Assert

Fails a test with the given message.

[**fail\(\)**](#) - Static method in class org.junit.Assert

Fails a test with no message.

[**Failure**](#) - Class in [org.junit.runner.notification](#)

A Failure holds a description of the failed test and the exception that was thrown while running it.

[**Failure\(Description, Throwable\)**](#) - Constructor for class

org.junit.runner.notification.Failure

Constructs a Failure with the given description and exception.

[**Filter**](#) - Class in [org.junit.runner.manipulation](#)

The canonical case of filtering is when you want to run a single test method in a class.

[**Filter\(\)**](#) - Constructor for class org.junit.runner.manipulation.Filter

[**filter\(Filter\)**](#) - Method in interface org.junit.runner.manipulation.Filterable

Remove tests that don't pass the parameter filter.

[**filter\(Filter\)**](#) - Method in class org.junit.runners.ParentRunner

[**Filterable**](#) - Interface in [org.junit.runner.manipulation](#)

Runners that allow filtering should implement this interface.

[**filterWith\(Filter\)**](#) - Method in class org.junit.runner.Request

Returns a Request that only contains those tests that should run when filter is applied

[**filterWith\(Description\)**](#) - Method in class org.junit.runner.Request

Returns a Request that only runs contains tests whose [Description](#) equals desiredDescription

[**fireTestAssumptionFailed\(Failure\)**](#) - Method in class

org.junit.runner.notification.RunNotifier

Invoke to tell listeners that an atomic test flagged that it assumed something false.

[**fireTestFailure\(Failure\)**](#) - Method in class

org.junit.runner.notification.RunNotifier

Invoke to tell listeners that an atomic test failed.

[**fireTestFinished\(Description\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Invoke to tell listeners that an atomic test finished.

[**fireTestIgnored\(Description\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Invoke to tell listeners that an atomic test was ignored.

[**fireTestRunFinished\(Result\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Do not invoke.

[**fireTestRunStarted\(Description\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Do not invoke.

[**fireTestStarted\(Description\)**](#) - Method in class

org.junit.runner.notification.[RunNotifier](#)

Invoke to tell listeners that an atomic test is about to start.

G

[**getActual\(\)**](#) - Method in error org.junit.[ComparisonFailure](#)

Returns the actual string value

[**getAnnotation\(Class<T>\)**](#) - Method in class org.junit.runner.[Description](#)

[**getAnnotations\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getChildren\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getChildren\(\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

[**getChildren\(\)**](#) - Method in class org.junit.runners.[Parameterized](#)

[**getChildren\(\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a list of objects that define the children of this Runner.

[**getChildren\(\)**](#) - Method in class org.junit.runners.[Suite](#)

[**getClassName\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getDescription\(\)**](#) - Method in interface org.junit.runner.[Describable](#)

[**getDescription\(\)**](#) - Method in class org.junit.runner.notification.[Failure](#)

[**getDescription\(\)**](#) - Method in class org.junit.runner.[Runner](#)

[**getDescription\(\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

[**getDisplayName\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getException\(\)**](#) - Method in class org.junit.runner.notification.[Failure](#)

[**getExpected\(\)**](#) - Method in error org.junit.[ComparisonFailure](#)

Returns the expected string value

[**getFailureCount\(\)**](#) - Method in class org.junit.runner.[Result](#)

[**getFailures\(\)**](#) - Method in class org.junit.runner.[Result](#)

[**getIgnoreCount\(\)**](#) - Method in class org.junit.runner.[Result](#)

[**getMessage\(\)**](#) - Method in error org.junit.[ComparisonFailure](#)

Returns "..." in place of common prefix and "..." in place of common suffix between expected and actual.

[**getMessage\(\)**](#) - Method in class org.junit.runner.notification.[Failure](#)

Convenience method

[**getMethodName\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getName\(\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a name used to describe this Runner

[**getRunCount\(\)**](#) - Method in class org.junit.runner.[Result](#)

[**getRunner\(RunnerBuilder, Class<?>\)**](#) - Method in class org.junit.runner.[Computer](#)

Create a single-class runner for testClass, using builder

[**getRunner\(\)**](#) - Method in class org.junit.runner.[Request](#)

Returns a [Runner](#) for this Request

[**getRunTime\(\)**](#) - Method in class org.junit.runner.[Result](#)

[**getSuite\(RunnerBuilder, Class<?>\[\]\)**](#) - Method in class org.junit.runner.[Computer](#)

Create a suite for classes, building Runners with builder.

[**getTestClass\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**getTestClass\(\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a TestClass object wrapping the class to be executed.

[**getTestHeader\(\)**](#) - Method in class org.junit.runner.notification.[Failure](#)

[**getTrace\(\)**](#) - Method in class org.junit.runner.notification.[Failure](#)

Convenience method

[**getVersion\(\)**](#) - Method in class org.junit.runner.[JUnitCore](#)



H

[**hashCode\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**hasItem\(T\)**](#) - Static method in class org.junit.matchers.[JUnitMatchers](#)

[**hasItem\(Matcher<? extends T>\)**](#) - Static method in class
org.junit.matchers.[JUnitMatchers](#)

[**hasItems\(T...\)**](#) - Static method in class org.junit.matchers.[JUnitMatchers](#)

[**hasItems\(Matcher<? extends T>...\)**](#) - Static method in class
org.junit.matchers.[JUnitMatchers](#)

I

[Ignore](#) - Annotation Type in [org.junit](#)

Sometimes you want to temporarily disable a test or a group of tests.

[instanceOf\(Class<?>\)](#) - Static method in class org.hamcrest.core.[IsInstanceOf](#)

Is the value an instance of a particular type?

[Is<T>](#) - Class in [org.hamcrest.core](#)

Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive.

[IsMatcher<T>](#) - Constructor for class org.hamcrest.core.[Is](#)

[isMatcher<T>](#) - Static method in class org.hamcrest.core.[Is](#)

Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive.

[is\(T\)](#) - Static method in class org.hamcrest.core.[Is](#)

This is a shortcut to the frequently used `is(equalTo(x))`.

[is\(Class<?>\)](#) - Static method in class org.hamcrest.core.[Is](#)

This is a shortcut to the frequently used `is(instanceOf(SomeClass.class))`.

[IsAnything<T>](#) - Class in [org.hamcrest.core](#)

A matcher that always returns `true`.

[IsAnything\(\)](#) - Constructor for class org.hamcrest.core.[IsAnything](#)

[IsAnything\(String\)](#) - Constructor for class org.hamcrest.core.[IsAnything](#)

[isEmpty\(\)](#) - Method in class org.junit.runner.[Description](#)

[isEqual<T>](#) - Class in [org.hamcrest.core](#)

Is the value equal to another value, as tested by the `Object.equals(java.lang.Object)` invokedMethod?

[isEqual\(T\)](#) - Constructor for class org.hamcrest.core.[IsEqual](#)

[IsInstanceOf](#) - Class in [org.hamcrest.core](#)

Tests whether the value is an instance of a class.

[IsInstanceOf\(Class<?>\)](#) - Constructor for class org.hamcrest.core.[IsInstanceOf](#)

Creates a new instance of IsInstanceOf

[IsNot<T>](#) - Class in [org.hamcrest.core](#)

Calculates the logical negation of a matcher.

[**IsNot**](#)([**Matcher**](#)<T>) - Constructor for class org.hamcrest.core.[**IsNot**](#)

[**IsNull**](#)<T> - Class in [org.hamcrest.core](#)

Is the value null?

[**IsNull\(\)**](#) - Constructor for class org.hamcrest.core.[**IsNull**](#)

[**IsSame**](#)<T> - Class in [org.hamcrest.core](#)

Is the value the same object as another value?

[**IsSame\(T\)**](#) - Constructor for class org.hamcrest.core.[**IsSame**](#)

[**isSuite\(\)**](#) - Method in class org.junit.runner.[**Description**](#)

[**isTest\(\)**](#) - Method in class org.junit.runner.[**Description**](#)

J

[JUnit4](#) - Class in [org.junit.runners](#)

Aliases the current default JUnit 4 class runner, for future-proofing.

[JUnit4\(Class<?>\)](#) - Constructor for class org.junit.runners.[JUnit4](#)

Constructs a new instance of the default runner

[JUnitCore](#) - Class in [org.junit.runner](#)

JUnitCore is a facade for running tests.

[JUnitCore\(\)](#) - Constructor for class org.junit.runner.[JUnitCore](#)

Create a new JUnitCore to run tests.

[JUnitMatchers](#) - Class in [org.junit.matchers](#)

Convenience import class: these are useful matchers for use with the assertEquals method, but they are not currently included in the basic CoreMatchers class from hamcrest.

[JUnitMatchers\(\)](#) - Constructor for class org.junit.matchers.[JUnitMatchers](#)

M

[**main\(String...\)**](#) - Static method in class org.junit.runner.[JUnitCore](#)

Run the tests contained in the classes named in the args.

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[AllOf](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[AnyOf](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[DescribedAs](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[Is](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[IsAnything](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[isEqual](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[IsInstanceOf](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[IsNot](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[IsNull](#)

[**matches\(Object\)**](#) - Method in class org.hamcrest.core.[IsSame](#)

[**matchMethodDescription>Description**](#) - Static method in class org.junit.runner.manipulation.[Filter](#)

Returns a Filter that only runs the single method described by desiredDescription

[**method\(Class<?>, String\)**](#) - Static method in class org.junit.runner.[Request](#)

Create a Request that, when processed, will run a single test.

[**methodBlock\(FrameworkMethod\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Returns a Statement that, when executed, either returns normally if method passes, or throws an exception if method fails.

[**methodInvoker\(FrameworkMethod, Object\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Returns a Statement that invokes method on test

N

[**not\(Matcher<T>\)**](#) - Static method in class org.hamcrest.core.[IsNot](#)

Inverts the rule.

[**not\(T\)**](#) - Static method in class org.hamcrest.core.[IsNot](#)

This is a shortcut to the frequently used not(equalTo(x)).

[**NoTestsRemainException**](#) - Exception in [org.junit.runner.manipulation](#)

Thrown when a filter removes all tests from a runner.

[**NoTestsRemainException\(\)**](#) - Constructor for exception

org.junit.runner.manipulation.[NoTestsRemainException](#)

[**notNullValue\(\)**](#) - Static method in class org.hamcrest.core.[IsNull](#)

Matches if value is not null.

[**notNullValue\(Class<T>\)**](#) - Static method in class org.hamcrest.core.[IsNull](#)

Matches if value is not null.

[**NULL**](#) - Static variable in class org.junit.runner.manipulation.[Sorter](#)

NULL is a Sorter that leaves elements in an undefined order

[**nullValue\(\)**](#) - Static method in class org.hamcrest.core.[IsNull](#)

Matches if value is null.

[**nullValue\(Class<T>\)**](#) - Static method in class org.hamcrest.core.[IsNull](#)

Matches if value is null.

O

[**org.hamcrest.core**](#) - package org.hamcrest.core

Fundamental matchers of objects and values, and composite matchers.

[**org.junit**](#) - package org.junit

Provides JUnit core classes and annotations.

[**org.junit.matchers**](#) - package org.junit.matchers

Provides useful additional Matchers for use with the

[`Assert.assertThat\(Object, org.hamcrest.Matcher\)`](#) statement

[**org.junit.runner**](#) - package org.junit.runner

Provides classes used to describe, collect, run and analyze multiple tests.

[**org.junit.runner.manipulation**](#) - package org.junit.runner.manipulation

Provides classes to [`filter`](#) or [`sort`](#) tests.

[**org.junit.runner.notification**](#) - package org.junit.runner.notification

Provides information about a test run.

[**org.junit.runners**](#) - package org.junit.runners

Provides standard [`Runner`](#) implementations.

P

[Parameterized](#) - Class in [org.junit.runners](#)

The custom runner Parameterized implements parameterized tests.

[Parameterized\(Class<?>\)](#) - Constructor for class

org.junit.runners.Parameterized

Only called reflectively.

[Parameterized.Parameters](#) - Annotation Type in [org.junit.runners](#)

Annotation for a method which provides parameters to be injected into the test class constructor by Parameterized

[ParentRunner<T>](#) - Class in [org.junit.runners](#)

Provides most of the functionality specific to a Runner that implements a "parent node" in the test tree, with children defined by objects of some data type T.

[ParentRunner\(Class<?>\)](#) - Constructor for class

org.junit.runners.ParentRunner

Constructs a new ParentRunner that will run @TestClass

[pleaseStop\(\)](#) - Method in class org.junit.runner.notification.RunNotifier

Ask that the tests run stop before starting the next test.

[possiblyExpectingExceptions\(FrameworkMethod, Object, Statement\)](#) -

Method in class org.junit.runners.BlockJUnit4ClassRunner

Deprecated. Will be private soon: use Rules instead

R

[**removeListener\(RunListener\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Remove a listener.

[**removeListener\(RunListener\)**](#) - Method in class
org.junit.runner.notification.[RunNotifier](#)

Internal use only

[**Request**](#) - Class in [org.junit.runner](#)

A Request is an abstract description of tests to be run.

[**Request\(\)**](#) - Constructor for class org.junit.runner.[Request](#)

[**Result**](#) - Class in [org.junit.runner](#)

A Result collects and summarizes information from running multiple tests.

[**Result\(\)**](#) - Constructor for class org.junit.runner.[Result](#)

[**Rule**](#) - Annotation Type in [org.junit](#)

Annotates fields that contain rules.

[**rules\(Object\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

[**run\(Class<?>...\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Run all the tests in classes.

[**run\(Computer, Class<?>...\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Run all the tests in classes.

[**run\(Request\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Run all the tests contained in request.

[**run\(Test\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Run all the tests contained in JUnit 3.8.x test.

[**run\(Runner\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Do not use.

[**run\(RunNotifier\)**](#) - Method in class org.junit.runner.[Runner](#)

Run the tests for this runner.

[**run\(RunNotifier\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

[**runChild\(FrameworkMethod, RunNotifier\)**](#) - Method in class
org.junit.runners.[BlockJUnit4ClassRunner](#)

[**runChild\(T, RunNotifier\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

Runs the test corresponding to child, which can be assumed to be an element of the list returned by [ParentRunner.getChildren\(\)](#).

[**runChild\(Runner, RunNotifier\)**](#) - Method in class org.junit.runners.[Suite](#)

[**runClasses\(Computer, Class<?>...\)**](#) - Static method in class org.junit.runner.[JUnitCore](#)

Run the tests contained in classes.

[**runClasses\(Class<?>...\)**](#) - Static method in class org.junit.runner.[JUnitCore](#)

Run the tests contained in classes.

[**RunListener**](#) - Class in [org.junit.runner.notification](#)

If you need to respond to the events during a test run, extend RunListener and override the appropriate methods.

[**RunListener\(\)**](#) - Constructor for class org.junit.runner.notification.[RunListener](#)

[**runMain\(JUnitSystem, String...\)**](#) - Method in class org.junit.runner.[JUnitCore](#)

Do not use.

[**runMainAndExit\(JUnitSystem, String...\)**](#) - Static method in class org.junit.runner.[JUnitCore](#)

Do not use.

[**runner\(Runner\)**](#) - Static method in class org.junit.runner.[Request](#)

[**Runner**](#) - Class in [org.junit.runner](#)

A Runner runs tests and notifies a [RunNotifier](#) of significant events as it does so.

[**Runner\(\)**](#) - Constructor for class org.junit.runner.[Runner](#)

[**RunNotifier**](#) - Class in [org.junit.runner.notification](#)

If you write custom runners, you may need to notify JUnit of your progress running tests.

[**RunNotifier\(\)**](#) - Constructor for class org.junit.runner.notification.[RunNotifier](#)

[**RunWith**](#) - Annotation Type in [org.junit.runner](#)

When a class is annotated with @RunWith or extends a class annotated with @RunWith, JUnit will invoke the class it references to run the tests in that class instead of the runner built into JUnit.

S

[**sameInstance\(T\)**](#) - Static method in class org.hamcrest.core.[IsSame](#)

Creates a new instance of IsSame

[**serial\(\)**](#) - Static method in class org.junit.runner.[Computer](#)

Returns a new default computer, which runs tests in serial order

[**setScheduler\(RunnerScheduler\)**](#) - Method in class

org.junit.runners.[ParentRunner](#)

Sets a scheduler that determines the order and parallelization of children.

[**shouldRun\(Description\)**](#) - Method in class org.junit.runner.manipulation.[Filter](#)

[**sort\(Sorter\)**](#) - Method in interface org.junit.runner.manipulation.[Sortable](#)

Sorts the tests using sorter

[**sort\(Sorter\)**](#) - Method in class org.junit.runners.[ParentRunner](#)

[**Sortable**](#) - Interface in [org.junit.runner.manipulation](#)

Interface for runners that allow sorting of tests.

[**Sorter**](#) - Class in [org.junit.runner.manipulation](#)

A Sorter orders tests.

[**Sorter\(Comparator<Description>\)**](#) - Constructor for class
org.junit.runner.manipulation.[Sorter](#)

Creates a Sorter that uses comparator to sort tests

[**sortWith\(Comparator<Description>\)**](#) - Method in class
org.junit.runner.[Request](#)

Returns a Request whose Tests can be run in a certain order, defined by
comparator For example, here is code to run a test suite in alphabetical
order:

[**StoppedByUserException**](#) - Exception in [org.junit.runner.notification](#)

Thrown when a user has requested that the test run stop.

[**StoppedByUserException\(\)**](#) - Constructor for exception
org.junit.runner.notification.[StoppedByUserException](#)

[**Suite**](#) - Class in [org.junit.runners](#)

Using Suite as a runner allows you to manually build a suite containing
tests from many classes.

[**Suite\(Class<?>, RunnerBuilder\)**](#) - Constructor for class org.junit.runners.[Suite](#)
Called reflectively on classes annotated with @RunWith(Suite.class)

[**Suite\(RunnerBuilder, Class<?>\[\]\)**](#) - Constructor for class org.junit.runners.Suite

Call this when there is no single root class (for example, multiple class names passed on the command line to [JUnitCore](#))

[**Suite\(Class<?>, Class<?>\[\]\)**](#) - Constructor for class org.junit.runners.Suite

Call this when the default builder is good enough.

[**Suite\(RunnerBuilder, Class<?>, Class<?>\[\]\)**](#) - Constructor for class org.junit.runners.Suite

Called by this class and subclasses once the classes making up the suite have been determined

[**Suite\(Class<?>, List<Runner>\)**](#) - Constructor for class org.junit.runners.Suite

Called by this class and subclasses once the runners making up the suite have been determined

[**Suite.SuiteClasses**](#) - Annotation Type in [org.junit.runners](#)

The SuiteClasses annotation specifies the classes to be run when a class annotated with @RunWith(Suite.class) is run.

T

[**Test**](#) - Annotation Type in [org.junit](#)

The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.

[**Test.None**](#) - Class in [org.junit](#)

Default empty exception

[**TEST_MECHANISM**](#) - Static variable in class org.junit.runner.[Description](#)

Describes a step in the test-running mechanism that goes so wrong no other description can be used (for example, an exception thrown from a Runner's constructor)

[**testAssumptionFailure\(Failure\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called when an atomic test flags that it assumes a condition that is false

[**testCount\(\)**](#) - Method in class org.junit.runner.[Description](#)

[**testCount\(\)**](#) - Method in class org.junit.runner.[Runner](#)

[**testFailure\(Failure\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called when an atomic test fails.

[**testFinished\(Description\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called when an atomic test has finished, whether the test succeeds or fails.

[**testIgnored\(Description\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called when a test will not be run, generally because a test method is annotated with [Ignore](#).

[**testName\(FrameworkMethod\)**](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Returns the name that describes method for [Descriptions](#).

[**testRunFinished\(Result\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called when all tests have finished

[**testRunStarted\(Description\)**](#) - Method in class org.junit.runner.notification.[RunListener](#)

Called before any tests have been run.

[**testStarted\(Description\)**](#) - Method in class

`org.junit.runner.notification.RunListener`

Called when an atomic test is about to be started.

toString() - Method in class `org.junit.runner.Description`

toString() - Method in class `org.junit.runner.notification.Failure`

V

[**validateConstructor\(List<Throwable>\)**](#) - Method in class
org.junit.runners.BlockJUnit4ClassRunner

Adds to errors if the test class has more than one constructor, or if the constructor takes parameters.

[**validateInstanceMethods\(List<Throwable>\)**](#) - Method in class
org.junit.runners.BlockJUnit4ClassRunner

Deprecated. *unused API, will go away in future version*

[**validateOnlyOneConstructor\(List<Throwable>\)**](#) - Method in class
org.junit.runners.BlockJUnit4ClassRunner

Adds to errors if the test class has more than one constructor (do not override)

[**validatePublicVoidNoArgMethods\(Class<? extends Annotation>, boolean, List<Throwable>\)**](#) - Method in class org.junit.runners.ParentRunner

Adds to errors if any method in this class is annotated with annotation, but: is not public, or takes parameters, or returns something other than void, or is static (given `isStatic` is `false`), or is not static (given `isStatic` is `true`).

[**validateTestMethods\(List<Throwable>\)**](#) - Method in class
org.junit.runners.BlockJUnit4ClassRunner

Adds to errors for each method annotated with `@Test` that is not a public, void instance method with no arguments.

[**validateZeroArgConstructor\(List<Throwable>\)**](#) - Method in class
org.junit.runners.BlockJUnit4ClassRunner

Adds to errors if the test class's single constructor takes parameters (do not override)

W

[wasSuccessful\(\)](#) - Method in class org.junit.runner.[Result](#)

[withAfterClasses\(Statement\)](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a Statement: run all non-overridden @AfterClass methods on this class and superclasses before executing statement; all AfterClass methods are always executed: exceptions thrown by previous steps are combined, if necessary, with exceptions from AfterClass methods into a MultipleFailureException.

[withAfters\(FrameworkMethod, Object, Statement\)](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Deprecated. *Will be private soon: use Rules instead*

[withBeforeClasses\(Statement\)](#) - Method in class org.junit.runners.[ParentRunner](#)

Returns a Statement: run all non-overridden @BeforeClass methods on this class and superclasses before executing statement; if any throws an Exception, stop execution and pass the exception on.

[withBefores\(FrameworkMethod, Object, Statement\)](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Deprecated. *Will be private soon: use Rules instead*

[withPotentialTimeout\(FrameworkMethod, Object, Statement\)](#) - Method in class org.junit.runners.[BlockJUnit4ClassRunner](#)

Deprecated. *Will be private soon: use Rules instead*

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [M](#) [N](#) [O](#) [P](#) [R](#) [S](#) [T](#) [V](#) [W](#)

[org.hamcrest.core](#)

[org.junit](#)

[org.junit.matchers](#)

[org.junit.runner](#)

[org.junit.runner.manipulation](#)

[org.junit.runner.notification](#)

[org.junit.runners](#)



... .

..

..

JUNIT API 4.7

JUNIT 4 API

JUnit 4	
org.hamcrest.core	Fundamental matchers of objects and values, and composite matchers.
org.junit	Provides JUnit core classes and annotations.
org.junit.matchers	Provides useful additional Matchers for use with the <code>Assert.assertThat(Object, org.hamcrest.Matcher)</code> statement
org.junit.runner	Provides classes used to describe, collect, run and analyze multiple tests.
org.junit.runner.manipulation	Provides classes to <code>filter</code> or <code>sort</code> tests.
org.junit.runner.notification	Provides information about a test run.
org.junit.runners	Provides standard <code>Runner</code> implementations.



Hierarchy For All Packages

Package Hierarchies:

[org.hamcrest.core](#), [org.junit](#), [org.junit.matchers](#), [org.junit.runner](#),
[org.junit.runner.manipulation](#), [org.junit.runner.notification](#), [org.junit.runners](#)

Class Hierarchy

- [java.lang.Object](#)
 - [org.junit.Assert](#)
 - [org.junit.Assume](#)
 - org.hamcrest.BaseMatcher<T> (implements org.hamcrest.Matcher<T>)
 - [org.hamcrest.core.AllOf<T>](#)
 - [org.hamcrest.core.AnyOf<T>](#)
 - [org.hamcrest.core.DescribedAs<T>](#)
 - [org.hamcrest.core.Is<T>](#)
 - [org.hamcrest.core.IsAnything<T>](#)
 - [org.hamcrest.core isEqual<T>](#)
 - [org.hamcrest.core IsInstanceOf](#)
 - [org.hamcrest.core IsNot<T>](#)
 - [org.hamcrest.coreIsNull<T>](#)
 - [org.hamcrest.core IsSame<T>](#)
 - [org.junit.runner.Computer](#)
 - [org.junit.runner.](#)
 - [org.junit.runner.notification.Failure](#)
 - [org.junit.runner.manipulation.Filter](#)
 - [org.junit.runner.JUnitCore](#)
 - [org.junit.matchers.JUnitMatchers](#)
 - [org.junit.runner.Request](#)
 - [org.junit.runner.Result](#)
 - [org.junit.runner.notification.RunListener](#)
 - [org.junit.runner.Runner](#) (implements [org.junit.runner.Describable](#))
 - org.junit.internal.runners.JUnit38ClassRunner (implements org.junit.runner.manipulation.Filterable, org.junit.runner.manipulation.Sortable)
 - [org.junit.internal.runners.SuiteMethod](#)
 - [org.junit.runners.AllTests](#)
 - [org.junit.runners.ParentRunner](#)<T> (implements org.junit.runner.manipulation.Filterable, org.junit.runner.manipulation.Sortable)
 - [org.junit.runners.BlockJUnit4ClassRunner](#)
 - [org.junit.runners.JUnit4](#)

- org.junit.runners.[Suite](#)
 - org.junit.runners.[Parameterized](#)
- org.junit.runner.notification.[RunNotifier](#)
- org.junit.runner.manipulation.[Sorter](#) (implements java.util.[Comparator](#)<T>)
- java.lang.[Throwable](#) (implements java.io.[Serializable](#))
 - java.lang.[Error](#)
 - java.lang.[AssertionError](#)
 - org.junit.[ComparisonFailure](#)
 - java.lang..
 - org.junit.runner.manipulation.[NoTestsRemainException](#)
 - java.lang.[RuntimeException](#)
 - org.junit.runner.notification.[StoppedByUserException](#)
 - org.junit.[Test.None](#)

Interface Hierarchy

- org.junit.runner.[**Describable**](#)
- org.junit.runner.manipulation.[**Filterable**](#)
- org.junit.runner.manipulation.[**Sortable**](#)

Annotation Type Hierarchy

- org.junit.[Test](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Rule](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Ignore](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[BeforeClass](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Before](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[AfterClass](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[After](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.runner.[RunWith](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.runners.[Suite.SuiteClasses](#) (implements
java.lang.annotation.[Annotation](#))
 - org.junit.runners.[Parameterized.Parameters](#) (implements
java.lang.annotation.[Annotation](#))
-
- 
-

.....

Serialized Form

org.junit

Class [org.junit.ComparisonFailure](#) extends [AssertionError](#) implements Serializable

serialVersionUID: 1L

Serialized Fields

fExpected

`String fExpected`

fActual

`String fActual`

Class [org.junit.Test.None](#) extends [Throwable](#) implements Serializable

serialVersionUID: 1L

org.junit.runner.manipulation

**Class
[org.junit.runner.manipulation.NoTestsRemainException](#) extends [Exception](#) implements Serializable**

serialVersionUID: 1L

org.junit.runner.notification

Class

[org.junit.runner.notification.StoppedByUserException](#)

extends [RuntimeException](#) implements Serializable

serialVersionUID: 1L

.....

.....



... 1 2 ...

... 1 2 ...



org.hamcrest.core Class AllOf<T>

```
java.lang.Object
└ org.hamcrest.BaseMatcher<T>
  └ org.hamcrest.core.AllOf<T>
```

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

```
public class AllOf<T>
```

```
extends org.hamcrest.BaseMatcher<T>
```

Calculates the logical conjunction of two matchers. Evaluation is shortcut, so that the second matcher is not called if the first matcher returns false.

```
AllOf(Iterable<org.hamcrest.Matcher<? extends T>> matchers)
```

static

<T> org.hamcrest.Matcher<T>

```
allOf(Iterable<org.hamcrest.Matcher<? extends T>> matchers)
```

Evaluates to true only if ALL of the passed in matchers evaluate to true.

```
static
<T> org.hamcrest.Matcher<T>
```

```
allOf(org.hamcrest.Matcher<? extends T>... matchers)
```

Evaluates to true only if ALL of the passed in matchers evaluate to true.

```
void describeTo(org.hamcrest.Description description)
```

Generates a description of the object. boolean matches(Object o)

Evaluates the matcher for argument *item*.

```
org.hamcrest.BaseMatcher
```

```
_dont_implement_Matcher____instead_extend_BaseMatcher_, toString
```

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

AllOf

public **AllOf**([Iterable](#)<org.hamcrest.Matcher<? extends [T](#)>> matchers)

matches

public boolean **matches**([Object](#) o)

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

o - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

public void **describeTo**(org.hamcrest.Description description)

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

allOf

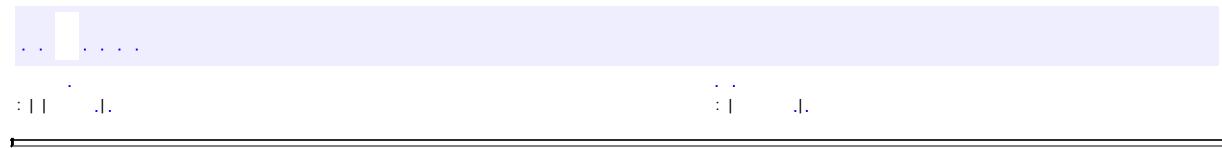
public static <T> org.hamcrest.Matcher<T> **allOf**(org.hamcrest.Matcher

Evaluates to true only if ALL of the passed in matchers evaluate to true.

allOf

public static <T> org.hamcrest.Matcher<T> **allOf**([Iterable](#)<org.hamcrest.Matcher>... matchers)

Evaluates to true only if ALL of the passed in matchers evaluate to true.



A screenshot of a Java code editor showing a line of code. The line starts with 'public static <T> org.hamcrest.Matcher<T> allOf(' followed by an Iterable type parameter. The argument is a list of matchers, indicated by three dots (...). A cursor is positioned after the closing parenthesis of the method call. The background of the code editor has a light purple gradient bar at the top.



... 1 ...

... 1 ...



org.hamcrest.core Class AnyOf<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ org.hamcrest.core.AnyOf<T>

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

public class AnyOf<T>

extends org.hamcrest.BaseMatcher<T>

Calculates the logical disjunction of two matchers. Evaluation is shortcut, so that the second matcher is not called if the first matcher returns true.

[AnyOf\(Iterable<org.hamcrest.Matcher<? extends I>> matchers\)](#)

[anyOf\(Iterable<org.hamcrest.Matcher<? extends T>> matchers\)](#)

Evaluates to true if ANY of the passed in matchers evaluate to true.

static
<T> org.hamcrest.Matcher<T>
[anyOf\(org.hamcrest.Matcher<? extends T>... matchers\)](#)
Evaluates to true if ANY of the passed in matchers evaluate to true. void
[describeTo\(org.hamcrest.Description description\)](#)
Generates a description of the object. boolean [matches\(Object o\)](#)
Evaluates the matcher for argument *item*.

org.hamcrest.BaseMatcher

_dont_implement_Matcher____instead_extend_BaseMatcher_, [toString](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

AnyOf

public **AnyOf**([Iterable](#)<org.hamcrest.Matcher<? extends [T](#)>> matchers)

matches

public boolean **matches**([Object](#) o)

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

o - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

public void **describeTo**(org.hamcrest.Description description)

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

anyOf

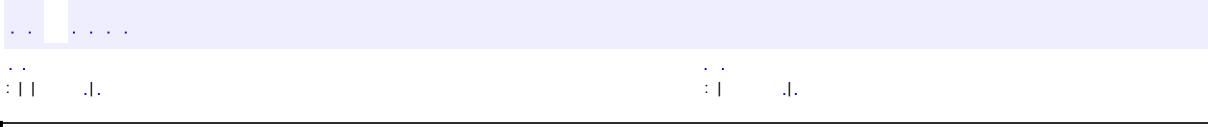
public static <T> org.hamcrest.Matcher<T> **anyOf**(org.hamcrest.Matcher

Evaluates to true if ANY of the passed in matchers evaluate to true.

anyOf

public static <T> org.hamcrest.Matcher<T> **anyOf**([Iterable](#)<org.hamcrest.Matcher<T>> matchers)

Evaluates to true if ANY of the passed in matchers evaluate to true.



The image shows a screenshot of a Java code editor. It features a light blue header bar with the word 'anyOf' in white. Below the header, there is a light gray workspace containing two lines of code. The first line starts with 'public static <T> org.hamcrest.Matcher<T> anyOf(' followed by a parameter list. The second line starts with 'Iterable<org.hamcrest.Matcher<T>> matchers)'. A blue cursor is positioned at the end of the second line. Above the workspace, there is a dark gray toolbar with several small, semi-transparent icons representing different code navigation and editing functions.



... 1 ...

... 1 ...



org.hamcrest.core Class DescribedAs<T>

```
java.lang.Object
└ org.hamcrest.BaseMatcher<T>
  └ org.hamcrest.core.DescribedAs<T>
```

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

```
public class DescribedAs<T>
```

```
extends org.hamcrest.BaseMatcher<T>
```

Provides a custom description to another matcher.

```
DescribedAs(String descriptionTemplate,
org.hamcrest.Matcher<T> matcher, Object[] values)
```

```
static
<T> org.hamcrest.Matcher<T>
describedAs(String description, org.hamcrest.Matcher<T> matcher,
Object... values)
```

Wraps an existing matcher and overrides the description when it fails.
void
[describeTo](#)(org.hamcrest.Description description)

Generates a description of the object. boolean [matches](#)(Object o)

Evaluates the matcher for argument item.

```
org.hamcrest.BaseMatcher
```

```
_dont_implement_Matcher____instead_extend_BaseMatcher_, toString
```

```
java.lang. Object
```

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
wait, wait, wait
```

DescribedAs

```
public DescribedAs(String descriptionTemplate,  
                    org.hamcrest.Matcher<T> matcher,  
                    Object[] values)
```

matches

```
public boolean matches(Object o)
```

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

- :
o - the object against which the matcher is evaluated.
- :
true if *item* matches, otherwise false.
- :
BaseMatcher

describeTo

```
public void describeTo(org.hamcrest.Description description)
```

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a a description of a larger object of which this is just a component, so it should

be worded appropriately.

:

description - The description to be built or appended to.

describedAs

```
public static <T> org.hamcrest.Matcher<T> describedAs(String descrip  
org.hamcrest.M  
Object... valu
```

Wraps an existing matcher and overrides the description when it fails.



The image shows a screenshot of a Java code editor. A tooltip is open over the `describedAs` method, displaying its signature: `public static <T> org.hamcrest.Matcher<T> describedAs(String descrip, Object... value)`. The code editor background is light blue, and the tooltip has a light purple header bar.



... 1 ...

... 1 ...



org.hamcrest.core Class Is<T>

```
java.lang.Object
└ org.hamcrest.BaseMatcher<T>
  └ org.hamcrest.core.Is<T>
```

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

```
public class Is<T>
  extends org.hamcrest.BaseMatcher<T>
```

Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive. eg. assertThat(cheese, equalTo(smelly)) vs assertThat(cheese, is(equalTo(smelly)))

Is (org.hamcrest.Matcher<T> matcher)	

void	describeTo (org.hamcrest.Description description) Generates a description of the object.
static org.hamcrest.Matcher<Object> is(Class<?> type)	This is a shortcut to the frequently used is(instanceOf(SomeClass.class)).

is(org.hamcrest.Matcher<T> matcher)

Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive.

```
static
<T> org.hamcrest.Matcher<T>
```

is(T value)

This is a shortcut to the frequently used is(equalTo(x)). boolean

matches(Object arg)

Evaluates the matcher for argument *item*.

org.hamcrest.BaseMatcher

`dont_implement_Matcher____instead_extend_BaseMatcher_, toString`

java.lang. Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Is

`public Is(org.hamcrest.Matcher<T> matcher)`

matches

`public boolean matches(Object arg)`

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

arg - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

```
public void describeTo(org.hamcrest.Description description)
```

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

is

```
public static <T> org.hamcrest.Matcher<T> is(org.hamcrest.Matcher<T>
```

Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive. eg. `assertThat(cheese, equalTo(smelly))` vs `assertThat(cheese, is(equalTo(smelly)))`

is

```
public static <T> org.hamcrest.Matcher<T> is(T value)
```

This is a shortcut to the frequently used `is(equalTo(x))`. eg. `assertThat(cheese, is(equalTo(smelly)))` vs `assertThat(cheese, is(smelly))`

is

```
public static org.hamcrest.Matcher<Object> is(Class<?> type)
```

This is a shortcut to the frequently used `is(instanceOf(SomeClass.class))`. eg. `assertThat(cheese, is(instanceOf(Cheddar.class)))` vs `assertThat(cheese, is(Cheddar.class))`



... 1.1.

... 1.1.



... 1 ...

... 1 ...



org.hamcrest.core Class IsAnything<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ **org.hamcrest.core.IsAnything<T>**

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

```
public class IsAnything<T>
extends org.hamcrest.BaseMatcher<T>
```

A matcher that always returns true.

[IsAnything\(\)](#)

[IsAnything\(String description\)](#)

static

<T> org.hamcrest.Matcher<T>

[any\(Class<T> type\)](#)

This matcher always evaluates to true.

```
static
<T> org.hamcrest.Matcher<T>
```

[anything\(\)](#)

This matcher always evaluates to true.

```
static
<T> org.hamcrest.Matcher<T>
```

[anything\(String description\)](#)

This matcher always evaluates to true. void

[describeTo\(org.hamcrest.Description description\)](#)

Generates a description of the object. boolean [matches\(Object o\)](#)
Evaluates the matcher for argument *item*.

org.hamcrest.BaseMatcher

[_dont_implement_Matcher__instead_extend_BaseMatcher_](#), [toString](#)

java.lang. Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

IsAnything

public **IsAnything()**

IsAnything

public **IsAnything(String description)**

matches

public boolean **matches(Object o)**

Description copied from interface: org.hamcrest.Matcher
Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

o - the object against which the matcher is evaluated.

```
:  
    true if item matches, otherwise false.  
:  
BaseMatcher
```

describeTo

```
public void describeTo(org.hamcrest.Description description)
```

Description copied from interface: org.hamcrest.SelfDescribing
Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

```
:  
description - The description to be built or appended to.
```

anything

```
public static <T> org.hamcrest.Matcher<T> anything()
```

This matcher always evaluates to true.

anything

```
public static <T> org.hamcrest.Matcher<T> anything(String description)
```

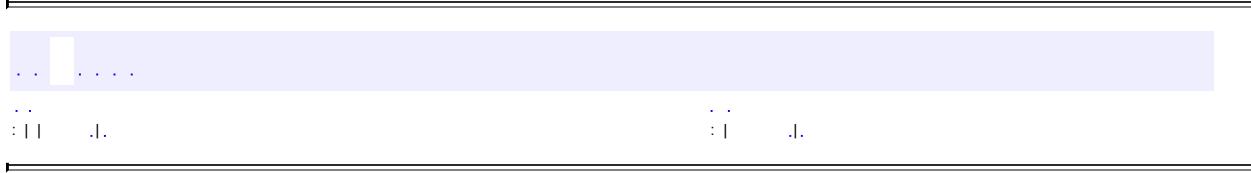
This matcher always evaluates to true.

```
:  
description - A meaningful string used when describing itself.
```

any

```
public static <T> org.hamcrest.Matcher<T> any(Class<T> type)
```

This matcher always evaluates to true. With type inference.





... 1 ...

... 1 ...



org.hamcrest.core Class IsEqual<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ **org.hamcrest.core.AreEqual<T>**

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

public class **.AreEqual<T>**

extends org.hamcrest.BaseMatcher<T>

Is the value equal to another value, as tested by the
[Object.equals\(java.lang.Object\)](#) invokedMethod?

[isEqual\(T equalArg\)](#)

void

[describeTo\(org.hamcrest.Description description\)](#)

Generates a description of the object.

static

<T>

org.hamcrest.Matcher<T>

[equalTo\(T operand\)](#)

Is the value equal to another value, as tested by the

[Object.equals\(java.lang.Object\)](#) invokedMethod? boolean

[matches\(Object arg\)](#)

Evaluates the matcher for argument *item*.

org.hamcrest.BaseMatcher

[_dont_implement_Matcher____instead_extend_BaseMatcher_, toString](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

isEqual

public [isEqual](#)([T](#) equalArg)

matches

public boolean [matches](#)([Object](#) arg)

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

arg - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

public void [describeTo](#)(org.hamcrest.Description description)

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

equalTo

public static <T> org.hamcrest.Matcher<T> **equalTo**(T operand)

Is the value equal to another value, as tested by the
[Object.equals\(java.lang.Object\)](#) invokedMethod?





... 1 ...

... 1 ...



org.hamcrest.core Class IsInstanceOf

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<[Object](#)>
 └ org.hamcrest.core.IsInstanceOf

:

org.hamcrest.Matcher<[Object](#)>, org.hamcrest.SelfDescribing

public class IsInstanceOf
extends org.hamcrest.BaseMatcher<[Object](#)>

Tests whether the value is an instance of a class.

IsInstanceOf(Class<?> theClass)	Creates a new instance of IsInstanceOf
---	--

IsInstanceOf(Class<?> theClass)	Creates a new instance of IsInstanceOf
void	describeTo(org.hamcrest.Description description) Generates a description of the object.
static org.hamcrest.Matcher< Object >	instanceOf(Class<?> type) Is the value an instance of a particular type?
boolean	matches(Object item) Evaluates the matcher for argument <i>item</i> .

org.hamcrest.BaseMatcher

don't_implement_Matcher____instead_extend_BaseMatcher_, toString

java.lang. Object

clone , equals , finalize , getClass , hashCode , notify , notifyAll , wait , wait , wait
--

IsInstanceOf

```
public IsInstanceOf<Class<?>> theClass)
```

Creates a new instance of IsInstanceOf

:

theClass - The predicate evaluates to true for instances of this class or one of its subclasses.

matches

```
public boolean matches(Object item)
```

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

item - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

```
public void describeTo(org.hamcrest.Description description)
```

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

instanceOf

```
public static org.hamcrest.Matcher<Object> instanceOf(Class<?> type)
```

Is the value an instance of a particular type?

A screenshot of a Java code editor showing the `instanceOf` method. The code is as follows:

```
public static org.hamcrest.Matcher<Object> instanceOf(Class<?> type)
```

The code editor has a light blue background with dark blue horizontal bars at the top and bottom. The code is written in a white font. The cursor is positioned after the opening brace of the method definition.



... 1 ...

... 1 ...



org.hamcrest.core Class IsNot<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ org.hamcrest.core.IsNot<T>

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

public class IsNot<T>

extends org.hamcrest.BaseMatcher<T>

Calculates the logical negation of a matcher.

[IsNot](#)(org.hamcrest.Matcher<I> matcher)

void	describeTo (org.hamcrest.Description description) Generates a description of the object.
------	---

boolean	matches (Object arg) Evaluates the matcher for argument <i>item</i> .
---------	--

static <T> org.hamcrest.Matcher<T>	
--	--

[not](#)(org.hamcrest.Matcher<T> matcher)

Inverts the rule.

static
<T> org.hamcrest.Matcher<T>
[not](#)(T value)

This is a shortcut to the frequently used not(equalTo(x)).

org.hamcrest.BaseMatcher

_dont_implement_Matcher____instead_extend_BaseMatcher_, toString

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

IsNot

public **IsNot**(org.hamcrest.Matcher<[T](#)> matcher)

matches

public boolean **matches**([Object](#) arg)

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

arg - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

public void **describeTo**(org.hamcrest.Description description)

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

not

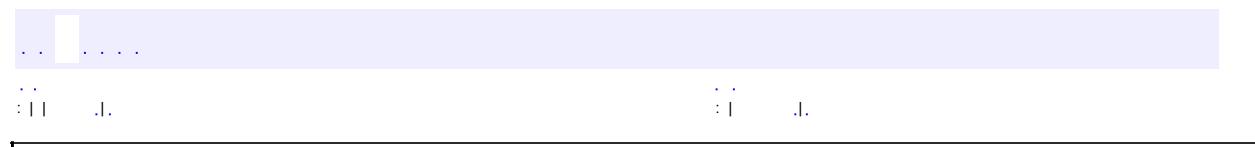
```
public static <T> org.hamcrest.Matcher<T> not(org.hamcrest.Matcher<T
```

Inverts the rule.

not

```
public static <T> org.hamcrest.Matcher<T> not(T value)
```

This is a shortcut to the frequently used not(equalTo(x)). eg.
assertThat(cheese, is(not(equalTo(smelly)))) vs assertThat(cheese,
is(not(smelly)))





... 1 ...

... 1 ...



org.hamcrest.core Class IsNull<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ org.hamcrest.core.IsNull<T>

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

public class IsNull<T>

extends org.hamcrest.BaseMatcher<T>

Is the value null?

[IsNull\(\)](#)

void	describeTo(org.hamcrest.Description description) Generates a description of the object.
------	--

boolean	matches(Object o) Evaluates the matcher for argument <i>item</i> .
---------	---

static <T> org.hamcrest.Matcher<T>	
--	--

[notNullValue\(\)](#)

Matches if value is not null.

static
<T> org.hamcrest.Matcher<T>

[notNullValue\(Class<T> type\)](#)

Matches if value is not null.

static

<T> org.hamcrest.Matcher<T>

[**nullValue\(\)**](#)

Matches if value is null.

static

<T> org.hamcrest.Matcher<T>

[**nullValue\(Class<T> type\)**](#)

Matches if value is null.

org.hamcrest.BaseMatcher

`_dont_implement_Matcher____instead_extend_BaseMatcher_, toString`

java.lang. Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

IsNull

public **IsNull()**

matches

public boolean **matches(Object o)**

Description copied from interface: org.hamcrest.Matcher

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

o - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

```
public void describeTo(org.hamcrest.Description description)
```

Description copied from interface: org.hamcrest.SelfDescribing

Generates a description of the object. The description may be part of a description of a larger object of which this is just a component, so it should be worded appropriately.

:

description - The description to be built or appended to.

nullValue

```
public static <T> org.hamcrest.Matcher<T> nullValue()
```

Matches if value is null.

notNullValue

```
public static <T> org.hamcrest.Matcher<T> notNullValue()
```

Matches if value is not null.

nullValue

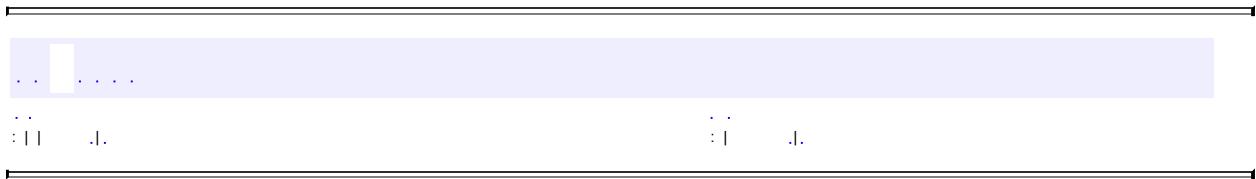
```
public static <T> org.hamcrest.Matcher<T> nullValue(Class<T> type)
```

Matches if value is null. With type inference.

notNullValue

```
public static <T> org.hamcrest.Matcher<T> notNullValue(Class<T> type
```

Matches if value is not null. With type inference.





CLASS

:| | :| |

CLASS

:| | :| |

org.hamcrest.core Class IsSame<T>

[java.lang.Object](#)
└ org.hamcrest.BaseMatcher<T>
 └ org.hamcrest.core.IsSame<T>

:

org.hamcrest.Matcher<T>, org.hamcrest.SelfDescribing

public class IsSame<T>

extends org.hamcrest.BaseMatcher<T>

Is the value the same object as another value?

[IsSame\(T object\)](#)

void	describeTo(org.hamcrest.Description description) Generates a description of the object.
------	--

boolean	matches(Object arg) Evaluates the matcher for argument <i>item</i> .
---------	---

static <T> org.hamcrest.Matcher<T>	
--	--

[sameInstance\(T object\)](#)

Creates a new instance of IsSame

org.hamcrest.BaseMatcher

[_dont_implement_Matcher_](#) [instead_extend_BaseMatcher_](#), [toString](#)

java.lang. Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),

[wait](#), [wait](#), [wait](#)

IsSame

public **IsSame**([T](#) object)

matches

public boolean **matches**([Object](#) arg)

Description copied from interface: [org.hamcrest.Matcher](#)

Evaluates the matcher for argument *item*.

This method matches against Object, instead of the generic type T. This is because the caller of the Matcher does not know at runtime what the type is (because of type erasure with Java generics). It is down to the implementations to check the correct type.

:

arg - the object against which the matcher is evaluated.

:

true if *item* matches, otherwise false.

:

BaseMatcher

describeTo

public void **describeTo**([org.hamcrest.Description](#) description)

Description copied from interface: [org.hamcrest.SelfDescribing](#)

Generates a description of the object. The description may be part of a a description of a larger object of which this is just a component, so it should

be worded appropriately.

:

description - The description to be built or appended to.

sameInstance

public static <T> org.hamcrest.Matcher<T> **sameInstance**(T object)

Creates a new instance of IsSame

:

object - The predicate evaluates to true only when the argument is this object.

A screenshot of a Java code editor showing the `sameInstance` method. The code is as follows:

```
public static <T> org.hamcrest.Matcher<T> sameInstance(T object)
```

The word "CLASS" is highlighted in blue. The code editor has a light purple background with horizontal scroll bars on the right and bottom.

[org.hamcrest.core](#) Àà [Allof](#)

[AnyOf](#)

[DescribedAs](#)

[Is](#)

[IsAnything](#)

[isEqual](#)

[IsInstanceOf](#)

[IsNot](#)

[IsNull](#)

[IsSame](#)



Package org.hamcrest.core

Fundamental matchers of objects and values, and composite matchers.

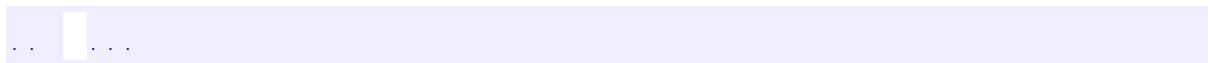
:

AllOf<T>	Calculates the logical conjunction of two matchers.
AnyOf<T>	Calculates the logical disjunction of two matchers.
DescribedAs<T>	Provides a custom description to another matcher.
Is<T>	Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive.
IsAnything<T>	A matcher that always returns true.
isEqual<T>	Is the value equal to another value, as tested by the <code>Object.equals(java.lang.Object)</code> invokedMethod?
IsInstanceOf	Tests whether the value is an instance of a class.
IsNot<T>	Calculates the logical negation of a matcher.
IsNull<T>	Is the value null?
IsSame<T>	Is the value the same object as another value?

Package org.hamcrest.core Description

Fundamental matchers of objects and values, and composite matchers.





[NEXT](#)



Hierarchy For Package org.hamcrest.core

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[Object](#)
 - org.hamcrest.BaseMatcher<T> (implements org.hamcrest.Matcher<T>)
 - org.hamcrest.core.[AllOf](#)<T>
 - org.hamcrest.core.[AnyOf](#)<T>
 - org.hamcrest.core.[DescribedAs](#)<T>
 - org.hamcrest.core.[Is](#)<T>
 - org.hamcrest.core.[IsAnything](#)<T>
 - org.hamcrest.core.[isEqual](#)<T>
 - org.hamcrest.core.[IsInstanceOf](#)
 - org.hamcrest.core.[IsNot](#)<T>
 - org.hamcrest.core.[IsNull](#)<T>
 - org.hamcrest.core.[IsSame](#)<T>

...  ...

[NEXT](#)



: REQUIRED | OPTIONAL

: ELEMENT



org.junit Annotation Type After

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface After
```

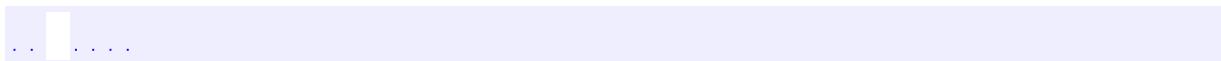
If you allocate external resources in a [Before](#) method you need to release them after the test runs. Annotating a public void method with @After causes that method to be run after the [Test](#) method. All @After methods are guaranteed to run even if a [Before](#) or [Test](#) method throws an exception. The @After methods declared in superclasses will be run after those of the current class.

Here is a simple example:

```
public class Example {
    File output;
    @Before public void createOutputFile() {
        output= new File(...);
    }
    @Test public void something() {
        ...
    }
    @After public void deleteOutputFile() {
        output.delete();
    }
}
```

:

[Before](#), [Test](#)



: REQUIRED | OPTIONAL

: ELEMENT



: REQUIRED | OPTIONAL

: ELEMENT



org.junit Annotation Type AfterClass

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface AfterClass
```

If you allocate expensive external resources in a [BeforeClass](#) method you need to release them after all the tests in the class have run. Annotating a public static void method with `@AfterClass` causes that method to be run after all the tests in the class have been run. All `@AfterClass` methods are guaranteed to run even if a [BeforeClass](#) method throws an exception. The `@AfterClass` methods declared in superclasses will be run after those of the current class.

Here is a simple example:

```
public class Example {
    private static DatabaseConnection database;
    @BeforeClass public static void login() {
        database= ...;
    }
    @Test public void something() {
        ...
    }
    @Test public void somethingElse() {
        ...
    }
    @AfterClass public static void logout() {
        database.logout();
    }
}
```

:

[BeforeClass](#), [Test](#)



: REQUIRED | OPTIONAL

: ELEMENT



... 1 ...

... 1 ...



org.junit Class Assert

```
java.lang.Object
└ org.junit.Assert
```

```
public class Assert
extends Object
```

A set of assertion methods useful for writing tests. Only failed assertions are recorded. These methods can be used directly: `Assert.assertEquals(...)`, however, they read better if they are referenced through static import:

```
import static org.junit.Assert.*;
...
assertEquals(...);
```

```
:
AssertionError
```

protected	Assert()
Protect constructor since it is a static only class	

static void	assertArrayEquals (byte[] expecteds, byte[] actuals) Asserts that two byte arrays are equal.
static void	assertArrayEquals (char[] expecteds, char[] actuals) Asserts that two char arrays are equal.
static void	assertArrayEquals (double[] expecteds, double[] actuals, double delta) Asserts that two double arrays are equal.
static void	assertArrayEquals (float[] expecteds, float[] actuals, float delta)

	Asserts that two float arrays are equal.
static void	assertArrayEquals (int[] expecteds, int[] actuals) Asserts that two int arrays are equal.
static void	assertArrayEquals (long[] expecteds, long[] actuals) Asserts that two long arrays are equal.
static void	assertArrayEquals (Object[] expecteds, Object[] actuals) Asserts that two object arrays are equal.
static void	assertArrayEquals (short[] expecteds, short[] actuals) Asserts that two short arrays are equal.
static void	assertArrayEquals (String message, byte[] expecteds, byte[] actuals) Asserts that two byte arrays are equal.
static void	assertArrayEquals (String message, char[] expecteds, char[] actuals) Asserts that two char arrays are equal.
static void	assertArrayEquals (String message, double[] expecteds, double[] actuals, double delta) Asserts that two double arrays are equal.
static void	assertArrayEquals (String message, float[] expecteds, float[] actuals, float delta) Asserts that two float arrays are equal.
static void	assertArrayEquals (String message, int[] expecteds, int[] actuals) Asserts that two int arrays are equal.
static void	assertArrayEquals (String message, long[] expecteds, long[] actuals) Asserts that two long arrays are equal.
static void	assertArrayEquals (String message, Object[] expecteds, Object[] actuals) Asserts that two object arrays are equal.
static void	assertArrayEquals (String message, short[] expecteds, short[] actuals) Asserts that two short arrays are equal.
static void	assertEquals (double expected, double actual) Deprecated. Use <i>assertEquals(double expected, double actual, double epsilon)</i> instead
	assertEquals (double expected, double actual,

static void	<code>double delta)</code> Asserts that two doubles or floats are equal to within a positive delta.
static void	<code>assertEquals(long expected, long actual)</code> Asserts that two longs are equal.
static void	<code>assertEquals(Object[] expecteds, Object[] actuals)</code> Deprecated. use <code>assertArrayEquals</code>
static void	<code>assertEquals(Object expected, Object actual)</code> Asserts that two objects are equal.
static void	<code>assertEquals(String message, double expected, double actual)</code> Deprecated. Use <code>assertEquals(String message, double expected, double actual, double epsilon)</code> instead
static void	<code>assertEquals(String message, double expected, double actual, double delta)</code> Asserts that two doubles or floats are equal to within a positive delta.
static void	<code>assertEquals(String message, long expected, long actual)</code> Asserts that two longs are equal.
static void	<code>assertEquals(String message, Object[] expecteds, Object[] actuals)</code> Deprecated. use <code>assertArrayEquals</code>
static void	<code>assertEquals(String message, Object expected, Object actual)</code> Asserts that two objects are equal.
static void	<code>assertFalse(boolean condition)</code> Asserts that a condition is false.
static void	<code>assertFalse(String message, boolean condition)</code> Asserts that a condition is false.
static void	<code>assertNotNull(Object object)</code> Asserts that an object isn't null.
static void	<code>assertNotNull(String message, Object object)</code> Asserts that an object isn't null.
static void	<code>assertNotSame(Object unexpected, Object actual)</code> Asserts that two objects do not refer to the same object.
	<code>assertNotSame(String message, Object unexpected,</code>

	<code>static void <u>Object</u> actual)</code>	Asserts that two objects do not refer to the same object.
static void	<code><u>assertNull</u>(Object object)</code>	Asserts that an object is null.
static void	<code><u>assertNull</u>(String message, <u>Object</u> object)</code>	Asserts that an object is null.
static void	<code><u>assertSame</u>(Object expected, <u>Object</u> actual)</code>	Asserts that two objects refer to the same object.
static void	<code><u>assertSame</u>(String message, <u>Object</u> expected, <u>Object</u> actual)</code>	Asserts that two objects refer to the same object.
static <T> void	<code><u>assertThat</u>(String reason, T actual, org.hamcrest.Matcher<T> matcher)</code>	Asserts that actual satisfies the condition specified by matcher.
static <T> void	<code><u>assertThat</u>(T actual, org.hamcrest.Matcher<T> matcher)</code>	Asserts that actual satisfies the condition specified by matcher.
static void	<code><u>assertTrue</u>(boolean condition)</code>	Asserts that a condition is true.
static void	<code><u>assertTrue</u>(String message, boolean condition)</code>	Asserts that a condition is true.
static void	<code><u>fail()</u></code>	Fails a test with no message.
static void	<code><u>fail</u>(String message)</code>	Fails a test with the given message.
java.lang. <u>Object</u>		
<code><u>clone</u>, <u>equals</u>, <u>finalize</u>, <u>getClass</u>, <u>hashCode</u>, <u>notify</u>, <u>notifyAll</u>, <u>toString</u>, <u>wait</u>, <u>wait</u>, <u>wait</u></code>		

Assert

`protected Assert()`

Protect constructor since it is a static only class



assertTrue

```
public static void assertTrue(String message,  
                           boolean condition)
```

Asserts that a condition is true. If it isn't it throws an [AssertionError](#) with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)
condition - condition to be checked

assertTrue

```
public static void assertTrue(boolean condition)
```

Asserts that a condition is true. If it isn't it throws an [AssertionError](#) without a message.

:

condition - condition to be checked

assertFalse

```
public static void assertFalse(String message,  
                           boolean condition)
```

Asserts that a condition is false. If it isn't it throws an [AssertionError](#) with the given message.

:

message - the identifying message for the [AssertionError](#) (null

okay)
condition - condition to be checked

assertFalse

`public static void assertFalse(boolean condition)`

Asserts that a condition is false. If it isn't it throws an [AssertionError](#) without a message.

:

condition - condition to be checked

fail

`public static void fail(String message)`

Fails a test with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)

:

[AssertionError](#)

fail

`public static void fail()`

Fails a test with no message.

assertEquals

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

Asserts that two objects are equal. If they are not, an [AssertionError](#) is thrown with the given message. If expected and actual are null, they are considered equal.

:

 message - the identifying message for the [AssertionError](#) (null okay)
 expected - expected value
 actual - actual value

assertEquals

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

Asserts that two objects are equal. If they are not, an [AssertionError](#) without a message is thrown. If expected and actual are null, they are considered equal.

:

 expected - expected value
 actual - the value to check against expected

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                 Object[] expecteds,  
                                 Object[] actuals)  
                                 throws org.junit.internal.ArrayCompar
```

Asserts that two object arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message. If expecteds and actuals are null, they are considered equal.

:

 message - the identifying message for the [AssertionError](#) (null okay)
 expecteds - Object array or array of arrays (multi-dimensional array)

with expected values.

actuals - Object array or array of arrays (multi-dimensional array)
with actual values

:

org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(Object[] expecteds,  
Object[] actuals)
```

Asserts that two object arrays are equal. If they are not, an [AssertionError](#) is thrown. If expected and actual are null, they are considered equal.

:

expecteds - Object array or array of arrays (multi-dimensional array)
with expected values

actuals - Object array or array of arrays (multi-dimensional array)
with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
        byte[] expecteds,  
        byte[] actuals)  
throws org.junit.internal.ArrayCompari
```

Asserts that two byte arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)

expecteds - byte array with expected values.

actuals - byte array with actual values

:

org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(byte[] expecteds,  
                                byte[] actuals)
```

Asserts that two byte arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

expecteds - byte array with expected values.
actuals - byte array with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                char[] expecteds,  
                                char[] actuals)  
throws org.junit.internal.ArrayCompari
```

Asserts that two char arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)
expecteds - char array with expected values.
actuals - char array with actual values

:

org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(char[] expecteds,  
                                char[] actuals)
```

Asserts that two char arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

expecteds - char array with expected values.
actuals - char array with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                  short[] expecteds,  
                                  short[] actuals)  
    throws org.junit.internal.ArrayCompa
```

Asserts that two short arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)

expecteds - short array with expected values.

actuals - short array with actual values

:

org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(short[] expecteds,  
                                    short[] actuals)
```

Asserts that two short arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

expecteds - short array with expected values.

actuals - short array with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                  int[] expecteds,  
                                  int[] actuals)
```

```
throws org.junit.internal.ArrayCompari
```

Asserts that two int arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)

expecteds - int array with expected values.

actuals - int array with actual values

:

org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(int[] expecteds,  
                                    int[] actuals)
```

Asserts that two int arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

expecteds - int array with expected values.

actuals - int array with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                    long[] expecteds,  
                                    long[] actuals)  
throws org.junit.internal.ArrayCompari
```

Asserts that two long arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)

expecteds - long array with expected values.

```
    actuals - long array with actual values  
:  
    org.junit.internal.ArrayComparisonFailure
```

assertArrayEquals

```
public static void assertArrayEquals(long[] expecteds,  
                                    long[] actuals)
```

Asserts that two long arrays are equal. If they are not, an [AssertionError](#) is thrown.

```
:  
    expecteds - long array with expected values.  
    actuals - long array with actual values
```

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                    double[] expecteds,  
                                    double[] actuals,  
                                    double delta)  
throws org.junit.internal.ArrayCompa
```

Asserts that two double arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

```
:  
    message - the identifying message for the AssertionError (null  
    okay)  
    expecteds - double array with expected values.  
    actuals - double array with actual values  
:  
    org.junit.internal.ArrayComparisonFailure
```

assertArrayEquals

```
public static void assertArrayEquals(double[] expecteds,
```

```
    double[] actuals,  
    double delta)
```

Asserts that two double arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

 expecteds - double array with expected values.
 actuals - double array with actual values

assertArrayEquals

```
public static void assertArrayEquals(String message,  
                                    float[] expecteds,  
                                    float[] actuals,  
                                    float delta)  
throws org.junit.internal.ArrayCompari
```

Asserts that two float arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

 message - the identifying message for the [AssertionError](#) (null okay)
 expecteds - float array with expected values.
 actuals - float array with actual values
:
 org.junit.internal.ArrayComparisonFailure

assertArrayEquals

```
public static void assertArrayEquals(float[] expecteds,  
                                    float[] actuals,  
                                    float delta)
```

Asserts that two float arrays are equal. If they are not, an [AssertionError](#) is thrown.

:

expecteds - float array with expected values.
actuals - float array with actual values

assertEquals

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

Asserts that two doubles or floats are equal to within a positive delta. If they are not, an [AssertionError](#) is thrown with the given message. If the expected value is infinity then the delta value is ignored. NaNs are considered equal: assertEquals(Double.NaN, Double.NaN, *) passes

:

message - the identifying message for the [AssertionError](#) (null okay)
expected - expected value
actual - the value to check against expected
delta - the maximum delta between expected and actual for which both numbers are still considered equal.

assertEquals

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

Asserts that two longs are equal. If they are not, an [AssertionError](#) is thrown.

:

expected - expected long value.
actual - actual long value

assertEquals

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

Asserts that two longs are equal. If they are not, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)
expected - long expected value.
actual - long actual value

assertEquals

[@Deprecated](#)

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

Deprecated. Use `assertEquals(double expected, double actual, double epsilon)` instead

assertEquals

[@Deprecated](#)

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

Deprecated. Use `assertEquals(String message, double expected, double actual, double epsilon)` instead

assertEquals

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

Asserts that two doubles or floats are equal to within a positive delta. If

they are not, an [AssertionError](#) is thrown. If the expected value is infinity then the delta value is ignored. NaNs are considered equal:

`assertEquals(Double.NaN, Double.NaN, *)` passes

2

expected - expected value

`actual` - the value to check against `expected`

`delta` - the maximum delta between expected and actual for which both numbers are still considered equal.

assertNotNull

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

Asserts that an object isn't null. If it is an [AssertionError](#) is thrown with the given message.

2

message - the identifying message for the [AssertionError](#) (null by default)

object - Object to check or null

assertNotNull

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

Asserts that an object isn't null. If it is an [AssertionError](#) is thrown.

1

object - Object to check or null

assertNull

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

Asserts that an object is null. If it is not, an [AssertionError](#) is thrown with the given message.

1

message - the identifying message for the [AssertionError](#) (null okay)
object - Object to check or null

assertNull

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

Asserts that an object is null. If it isn't an [AssertionError](#) is thrown.

2

object - Object to check or null

assertSame

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

Asserts that two objects refer to the same object. If they are not, an [AssertionError](#) is thrown with the given message.

2

`message` - the identifying message for the [AssertionError](#) (null okay)
`expected` - the expected object
`actual` - the object to compare to expected

assertSame

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

Asserts that two objects refer to the same object. If they are not the same, an [AssertionError](#) without a message is thrown.

:

expected - the expected object
actual - the object to compare to expected

assertNotSame

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

Asserts that two objects do not refer to the same object. If they do refer to the same object, an [AssertionError](#) is thrown with the given message.

:

message - the identifying message for the [AssertionError](#) (null okay)
unexpected - the object you don't expect
actual - the object to compare to unexpected

assertNotSame

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

Asserts that two objects do not refer to the same object. If they do refer to the same object, an [AssertionError](#) without a message is thrown.

:

unexpected - the object you don't expect
actual - the object to compare to unexpected

assertEquals

[@Deprecated](#)

```
public static void assertEquals(String message,  
                             Object[] expecteds,  
                             Object[] actuals)
```

Deprecated. use `assertArrayEquals`

Asserts that two object arrays are equal. If they are not, an [AssertionError](#) is thrown with the given message. If expecteds and actuals are null, they are considered equal.

2

message - the identifying message for the [AssertionError](#) (null okay)

expecteds - Object array or array of arrays (multi-dimensional array) with expected values.

`actuals` - Object array or array of arrays (multi-dimensional array) with actual values

assertEquals

@Deprecated

Deprecated. use `assertArrayEquals`

Asserts that two object arrays are equal. If they are not, an [AssertionError](#) is thrown. If expected and actual are null, they are considered equal.

2

`expecteds` - Object array or array of arrays (multi-dimensional array) with expected values

`actuals` - Object array or array of arrays (multi-dimensional array) with actual values

assertThat

Asserts that actual satisfies the condition specified by matcher. If not, an [AssertionError](#) is thrown with information about the matcher and failing value. Example:

```
assertThat(0, is(1)); // fails:  
    // failure message:  
    // expected: is <1>  
    // got value: <0>  
assertThat(0, is(not(1))) // passes
```

Type Parameters:

- T - the static type accepted by the matcher (this can flag obvious compile-time problems such as `assertThat(1, is("a"))`)
 - :
 - actual - the computed value being compared
 - matcher - an expression, built of Matchers, specifying allowed values
 - :
 - CoreMatchers, [JUnitMatchers](#)
-

assertThat

```
public static <T> void assertThat(String reason,  
                                T actual,  
                                org.hamcrest.Matcher<T> matcher)
```

Asserts that actual satisfies the condition specified by matcher. If not, an [AssertionError](#) is thrown with the reason and information about the matcher and failing value. Example:

```
:  
    assertThat("Help! Integers don't work", 0, is(1)); // fails:  
        // failure message:  
        // Help! Integers don't work  
        // expected: is <1>  
        // got value: <0>  
    assertThat("Zero is one", 0, is(not(1))) // passes
```

Type Parameters:

- T - the static type accepted by the matcher (this can flag obvious compile-time problems such as `assertThat(1, is("a"))`)

:

- reason - additional information about the error
- actual - the computed value being compared
- matcher - an expression, built of Matchers, specifying allowed values

:

- CoreMatchers, [JUnitMatchers](#)





... 1 ...

... 1 ...



org.junit Class Assume

```
java.lang.Object
└ org.junit.Assume
```

```
public class Assume
    extends Object
```

A set of methods useful for stating assumptions about the conditions in which a test is meaningful. A failed assumption does not mean the code is broken, but that the test provides no useful information. The default JUnit runner treats tests with failing assumptions as ignored. Custom runners may behave differently. For example:

```
// only provides information if database is reachable.
@Test public void calculateTotalSalary() {
    DBConnection dbc = Database.connect();
    assumeNotNull(dbc);
    // ...
}
```

These methods can be used directly: `Assume.assertTrue(...)`, however, they read better if they are referenced through static import:

```
import static org.junit.Assume.*;
...
assertTrue(...);
```

Assume()	
----------	--

static void	assumeNoException(Throwable t)
-------------	--------------------------------

static void	assumeNoException(Throwable t)
-------------	--------------------------------

Use to assume that an operation completes normally.

	assumeNotNull(Object... objects)
static void	If called with one or more null elements in objects, the test will halt and be ignored.
static <T> void	

[assumeThat\(T actual, org.hamcrest.Matcher<T> matcher\)](#)

Call to assume that actual satisfies the condition specified by matcher.

[assumeTrue\(boolean b\)](#)

If called with an expression evaluating to false, the test will halt and be ignored.

java.lang. [Object](#)

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Assume

public Assume()

assumeTrue

public static void assumeTrue(boolean b)

If called with an expression evaluating to false, the test will halt and be ignored.

:

b -

assumeNotNull

public static void assumeNotNull(Object... objects)

If called with one or more null elements in objects, the test will halt and be ignored.

:

objects -

assumeThat

```
public static <T> void assumeThat(T actual,  
                                org.hamcrest.Matcher<T> matcher)
```

Call to assume that `actual` satisfies the condition specified by `matcher`. If not, the test halts and is ignored. Example:

:

```
assumeThat(1, is(1)); // passes  
foo(); // will execute  
assumeThat(0, is(1)); // assumption failure! test halts  
int x = 1 / 0; // will never execute
```

Type Parameters:

`T` - the static type accepted by the matcher (this can flag obvious compile-time problems such as `assumeThat(1, is("a"))`)

:

`actual` - the computed value being compared

`matcher` - an expression, built of Matchers, specifying allowed values

:

`CoreMatchers`, [JUnitMatchers](#)

assumeNoException

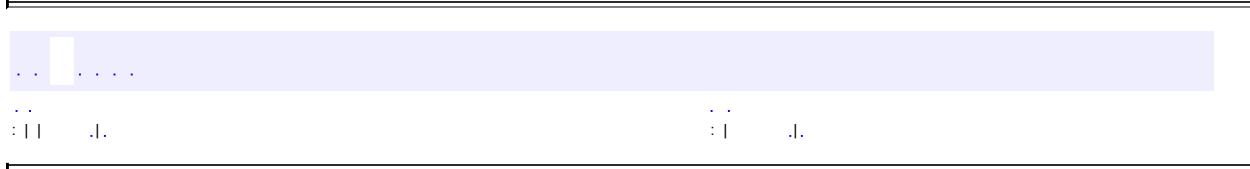
```
public static void assumeNoException(Throwable t)
```

Use to assume that an operation completes normally. If `t` is non-null, the test will halt and be ignored. For example:

```
@Test public void parseDataFile() {  
    DataFile file;  
    try {
```

```
        file = DataFile.open("sampledata.txt");
    } catch (IOException e) {
        // stop test and ignore if data can't be opened
        assumeNoException(e);
    }
    // ...
}

:
t - if non-null, the offending exception
```





: REQUIRED | OPTIONAL

: ELEMENT



org.junit Annotation Type Before

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface Before
```

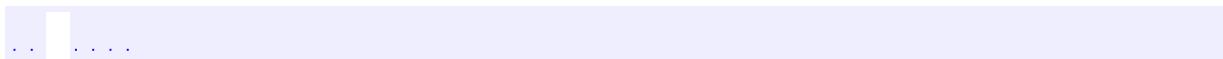
When writing tests, it is common to find that several tests need similar objects created before they can run. Annotating a `public void` method with `@Before` causes that method to be run before the [Test](#) method. The `@Before` methods of superclasses will be run before those of the current class. No other ordering is defined.

Here is a simple example:

```
public class Example {
    List empty;
    @Before public void initialize() {
        empty= new ArrayList();
    }
    @Test public void size() {
        ...
    }
    @Test public void remove() {
        ...
    }
}
```

:

[BeforeClass, After](#)



: REQUIRED | OPTIONAL : ELEMENT



: REQUIRED | OPTIONAL

: ELEMENT



org.junit Annotation Type BeforeClass

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface BeforeClass
```

Sometimes several tests need to share computationally expensive setup (like logging into a database). While this can compromise the independence of tests, sometimes it is a necessary optimization. Annotating a public static void no-arg method with `@BeforeClass` causes it to be run once before any of the test methods in the class. The `@BeforeClass` methods of superclasses will be run before those of the current class.

For example:

```
public class Example {
    @BeforeClass public static void onlyOnce() {
        ...
    }
    @Test public void one() {
        ...
    }
    @Test public void two() {
        ...
    }
}

:
```

[AfterClass](#)



: REQUIRED | OPTIONAL

: ELEMENT





... 1 ...

... 1 ...



org.junit Class ComparisonFailure

```
java.lang.Object
└ java.lang.Throwable
  └ java.lang.Error
    └ java.lang.AssertionError
      └ org.junit.ComparisonFailure

:
```

[Serializable](#)

```
public class ComparisonFailure
```

```
extends AssertionError
```

Thrown when an [assertEquals\(String, String\)](#) fails. Create and throw a ComparisonFailure manually if you want to show users the difference between two complex strings. Inspired by a patch from Alex Chaffee (alex@purpletech.com)

```
:
```

[Serialized Form](#)

comparisonFailure(String message, String expected, String actual) Constructs a comparison failure.		

String	getActual() Returns the actual string value	
String	getExpected() Returns the expected string value	
String	getMessage() Returns "..." in place of common prefix and "..." in place of common suffix between expected and actual.	

java.lang. [Throwable](#)

[fillInStackTrace](#), [getCause](#), [getLocalizedMessage](#), [getStackTrace](#),
[initCause](#), [printStackTrace](#), [printStackTrace](#), [printStackTrace](#),
[setStackTrace](#), [toString](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

ComparisonFailure

```
public ComparisonFailure(String message,  
                         String expected,  
                         String actual)
```

Constructs a comparison failure.

:

message - the identifying message or null
expected - the expected string value
actual - the actual string value

getMessage

```
public String getMessage()
```

Returns "..." in place of common prefix and "..." in place of common suffix between expected and actual.

:

[getMessage](#) in class [Throwable](#)

:

[Throwable.getMessage\(\)](#)

getActual

public [String](#) **getActual()**

Returns the actual string value

:

the actual string value

getExpected

public [String](#) **getExpected()**

Returns the expected string value

:

the expected string value





: REQUIRED | [OPTIONAL](#)

[ELEMENT](#)



org.junit Annotation Type Ignore

```
@Retention(value=RUNTIME)
@Target(value={METHOD,TYPE})
public @interface Ignore
```

Sometimes you want to temporarily disable a test or a group of tests. Methods annotated with [Test](#) that are also annotated with `@Ignore` will not be executed as tests. Also, you can annotate a class containing test methods with `@Ignore` and none of the containing tests will be executed. Native JUnit 4 test runners should report the number of ignored tests along with the number of tests that ran and the number of tests that failed.

For example:

```
@Ignore @Test public void something() { ... }
```

`@Ignore` takes an optional default parameter if you want to record why a test is being ignored:

```
@Ignore("not ready yet") @Test public void something() { ... }
```

`@Ignore` can also be applied to the test class:

```
@Ignore public class IgnoreMe {
    @Test public void test1() { ... }
    @Test public void test2() { ... }
}
```

Optional Element Summary

String	value
--------	--------------

 The optional reason why the test is ignored. |

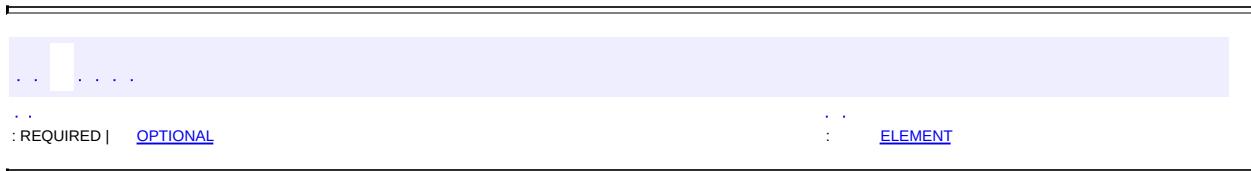
value

```
public abstract String value
```

The optional reason why the test is ignored.

:

""



[org.junit](#)

Àà [Assert](#)

[Assume](#)

[Test.None](#)

Errors

[ComparisonFailure](#)

Annotation Types

[After](#)

[AfterClass](#)

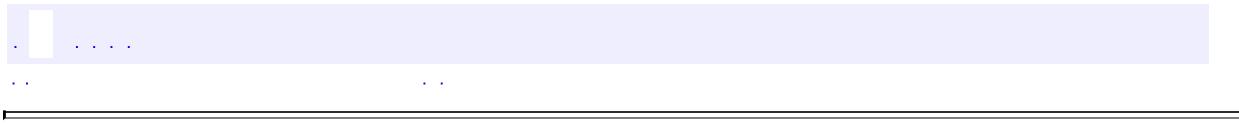
[Before](#)

[BeforeClass](#)

[Ignore](#)

[Rule](#)

[Test](#)



Package org.junit

Provides JUnit core classes and annotations.

:

Assert	A set of assertion methods useful for writing tests.
Assume	A set of methods useful for stating assumptions about the conditions in which a test is meaningful.
Test.None	Default empty exception

Error Summary

[ComparisonFailure](#) Thrown when an `assertEquals(String, String)` fails.

Annotation Types Summary

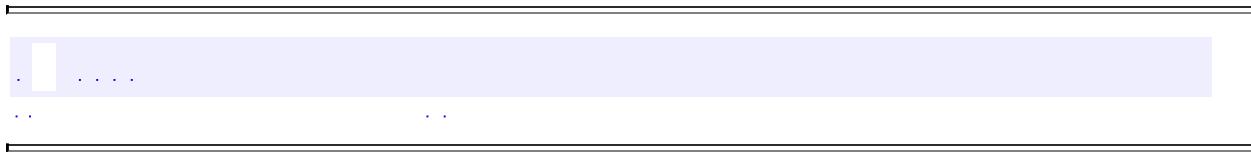
After	If you allocate external resources in a Before method you need to release them after the test runs.
AfterClass	If you allocate expensive external resources in a BeforeClass method you need to release them after all the tests in the class have run.
Before	When writing tests, it is common to find that several tests need similar objects created before they can run.
BeforeClass	Sometimes several tests need to share computationally expensive setup (like logging into a database).
Ignore	Sometimes you want to temporarily disable a test or a group of tests.
Rule	Annotates fields that contain rules.
Test	The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.

Package org.junit Description

Provides JUnit core classes and annotations. Corresponds to junit.framework in Junit 3.x.

:

4.0



...

...

...

...



[PREV](#) [NEXT](#)



Hierarchy For Package org.junit

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[**Object**](#)
 - org.junit.[**Assert**](#)
 - org.junit.[**Assume**](#)
 - java.lang.[**Throwable**](#) (implements java.io.[**Serializable**](#))
 - java.lang.[**Error**](#)
 - java.lang.[**AssertionError**](#)
 - org.junit.[**ComparisonFailure**](#)
 - org.junit.[**Test.None**](#)

Annotation Type Hierarchy

- org.junit.[Test](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Rule](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Ignore](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[BeforeClass](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[Before](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[AfterClass](#) (implements java.lang.annotation.[Annotation](#))
 - org.junit.[After](#) (implements java.lang.annotation.[Annotation](#))
-



[PREV](#) [NEXT](#)



: REQUIRED | OPTIONAL

: ELEMENT



org.junit Annotation Type Rule

```
@Retention(value=RUNTIME)
public @interface Rule
```

Annotates fields that contain rules. Such a field must be public, not static, and a subtype of `MethodRule`. For more information, see `MethodRule`





: REQUIRED | [OPTIONAL](#)

[ELEMENT](#)



org.junit Annotation Type Test

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public @interface Test
```

The `Test` annotation tells JUnit that the `public void` method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded.

A simple test looks like this:

```
public class Example {
    @Test
    public void method() {
        org.junit.Assert.assertTrue( new ArrayList().isEmpty() );
    }
}
```

The `Test` annotation supports two optional parameters. The first, `expected`, declares that a test method should throw an exception. If it doesn't throw an exception or if it throws a different exception than the one declared, the test fails. For example, the following test succeeds:

```
@Test(expected=IndexOutOfBoundsException.class) public void outOfBounds() {
    new ArrayList<Object>().get(1);
}
```

The second optional parameter, `timeout`, causes a test to fail if it takes longer than a specified amount of clock time (measured in milliseconds). The following test fails:

```
@Test(timeout=100) public void infinity() {
    while(true);
}
```

Optional Element Summary

<code>Class<? extends Throwable></code>	expected Optionally specify expected, a Throwable, to cause a test method to succeed iff an exception of the specified class is thrown by the method.
<code>long</code>	timeout Optionally specify timeout in milliseconds to cause a test method to fail if it takes longer than that number of milliseconds.

expected

```
public abstract Class<? extends Throwable> expected
```

Optionally specify expected, a Throwable, to cause a test method to succeed iff an exception of the specified class is thrown by the method.

:

```
org.junit.Test.None.class
```

timeout

```
public abstract long timeout
```

Optionally specify timeout in milliseconds to cause a test method to fail if it takes longer than that number of milliseconds.

:

```
0L
```

: REQUIRED | [OPTIONAL](#)

[ELEMENT](#)



CLASS

:|||

CLASS

:|||

org.junit Class Test.None

[java.lang.Object](#)
└ [java.lang.Throwable](#)
 └ [org.junit.Test.None](#)

:

[Serializable](#)

:

[Test](#)

public static class **Test.None**

extends [Throwable](#)

Default empty exception

:

[Serialized Form](#)

java.lang. [Throwable](#)

[fillInStackTrace](#), [getCause](#), [getLocalizedMessage](#), [getMessage](#),
[getStackTrace](#), [initCause](#), [printStackTrace](#), [printStackTrace](#),
[printStackTrace](#), [setStackTrace](#), [toString](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

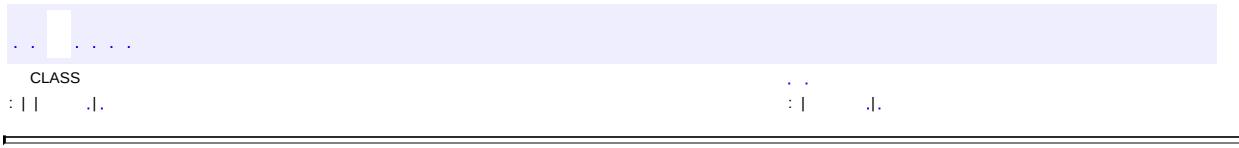
...

.. CLASS

:|:|: . . .

..

:|:|:



org.junit.matchers Class JUnitMatchers

```
java.lang.Object
└ org.junit.matchers.JUnitMatchers
```

```
public class JUnitMatchers
extends Object
```

Convenience import class: these are useful matchers for use with the assertThat method, but they are not currently included in the basic CoreMatchers class from hamcrest.

```
JUnitMatchers()
```

```
static
```

```
<T> org.junit.internal.matchers.CombinableMatcher<T>
```

```
both(org.hamcrest.Matcher<T> matcher)
```

This is useful for fluently combining matchers that must both pass.

```
static org.hamcrest.Matcher<String> containsString(String substring)
```

```
static
<T> org.junit.internal.matchers.CombinableMatcher<T>
either(org.hamcrest.Matcher<T> matcher)
```

This is useful for fluently combining matchers where either may pass, for example:

```
static
<T> org.hamcrest.Matcher<Iterable<T>>
everyItem(org.hamcrest.Matcher<T> elementMatcher)
```

```
static
<T> org.hamcrest.Matcher<Iterable<T>>
```

```
hasItem(org.hamcrest.Matcher<? extends T> elementMatcher)

static
<T> org.hamcrest.Matcher<Iterable<T>>
hasItem(T element)

static
<T> org.hamcrest.Matcher<Iterable<T>>
hasItems(org.hamcrest.Matcher<? extends T>... elementMatchers)

static
<T> org.hamcrest.Matcher<Iterable<T>>
hasItems(T... elements)
```

java.lang. [Object](#)

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

JUnitMatchers

```
public JUnitMatchers()
```

hasItem

```
public static <T> org.hamcrest.Matcher<Iterable<T>> hasItem(T elemen
:
    element -
:
A matcher matching any collection containing element
```

hasItem

```
public static <T> org.hamcrest.Matcher<Iterable<T>> hasItem(org.hamc  
:  
    elementMatcher -  
:  
        A matcher matching any collection containing an element matching  
        elementMatcher
```

hasItems

```
public static <T> org.hamcrest.Matcher<Iterable<T>> hasItems(T... el  
:  
    elements -  
:  
        A matcher matching any collection containing every element in  
        elements
```

hasItems

```
public static <T> org.hamcrest.Matcher<Iterable<T>> hasItems(org.ham  
:  
    elementMatchers -  
:  
        A matcher matching any collection containing at least one element that  
        matches each matcher in elementMatcher (this may be one element  
        matching all matchers, or different elements matching each matcher)
```

everyItem

```
public static <T> org.hamcrest.Matcher<Iterable<T>> everyItem(org.ha  
:  
    elementMatcher -  
:  
        A matcher matching any collection in which every element matches
```

elementMatcher

containsString

```
public static org.hamcrest.Matcher<String> containsString(String sub
:
    substring -
:
    a matcher matching any string that contains substring
```

both

```
public static <T> org.junit.internal.matchers.CombinableMatcher<T> b
```

This is useful for fluently combining matchers that must both pass. For example:

```
assertThat(string, both(containsString("a")).and(containsStri
```

either

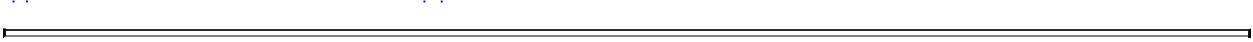
```
public static <T> org.junit.internal.matchers.CombinableMatcher<T> e
```

This is useful for fluently combining matchers where either may pass, for example:

```
assertThat(string, both(containsString("a")).and(containsStri
```



[org.junit.matchers](#) Àà [JUnitMatchers](#)



Package org.junit.matchers

Provides useful additional Matchers for use with the
[`Assert.assertThat\(Object, org.hamcrest.Matcher\)`](#) statement

:

JUnitMatchers	Convenience import class: these are useful matchers for use with the assertThat method, but they are not currently included in the basic CoreMatchers class from hamcrest.

Package org.junit.matchers Description

Provides useful additional Matchers for use with the
[`Assert.assertThat\(Object, org.hamcrest.Matcher\)`](#) statement

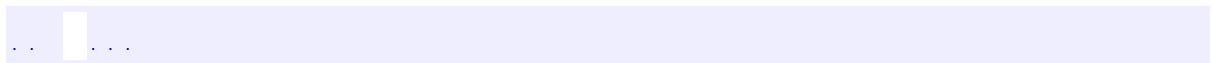
:

4.0

:

JUnitMatchers





[PREV](#) [NEXT](#)



Hierarchy For Package org.junit.matchers

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[Object](#)
 - org.junit.matchers.[JUnitMatchers](#)
-
- 
- ... PREV NEXT ...
-



... 1 2 ...

... 1 2 ...



org.junit.runner Class Computer

[java.lang.Object](#)
└ org.junit.runner.Computer

public class Computer

extends [Object](#)

Represents a strategy for computing runners and suites. WARNING: this class is very likely to undergo serious changes in version 4.8 and beyond.

[Computer\(\)](#)

protected Runner	getRunner (org.junit.runners.model.RunnerBuilder builder, Class <?> testClass) Create a single-class runner for testClass, using builder
Runner	getSuite (org.junit.runners.model.RunnerBuilder builder, Class <?>[] classes) Create a suite for classes, building Runners with builder.
static Computer	serial () Returns a new default computer, which runs tests in serial order

[java.lang. Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)



Computer

```
public Computer()
```



serial

```
public static Computer serial()
```

Returns a new default computer, which runs tests in serial order

getSuite

```
public Runner getSuite(org.junit.runners.model.RunnerBuilder builder  
                      Class<?>[] classes)  
    throws org.junit.runners.model.InitializationError
```

Create a suite for classes, building Runners with builder. Throws an InitializationError if Runner construction fails

:

```
org.junit.runners.model.InitializationError
```



getRunner

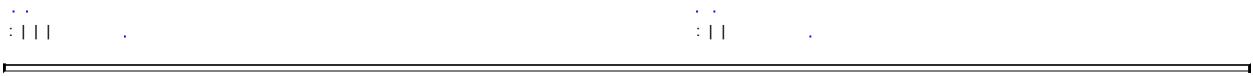
```
protected Runner getRunner(org.junit.runners.model.RunnerBuilder bui  
                           Class<?> testClass)  
    throws Throwable
```

Create a single-class runner for testClass, using builder

:

```
Throwable
```





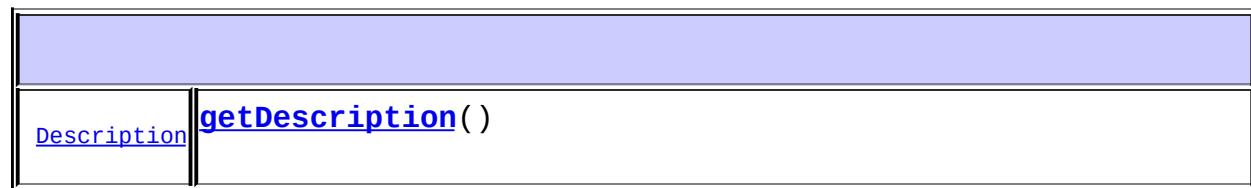
org.junit.runner Interface Describable

All Known Implementing Classes:

[AllTests](#), [BlockJUnit4ClassRunner](#),
[org.junit.internal.runners.JUnit38ClassRunner](#), [JUnit4](#), [Parameterized](#),
[ParentRunner](#), [Runner](#), [Suite](#), [org.junit.internal.runners.SuiteMethod](#)

public interface **Describable**

Represents an object that can describe itself

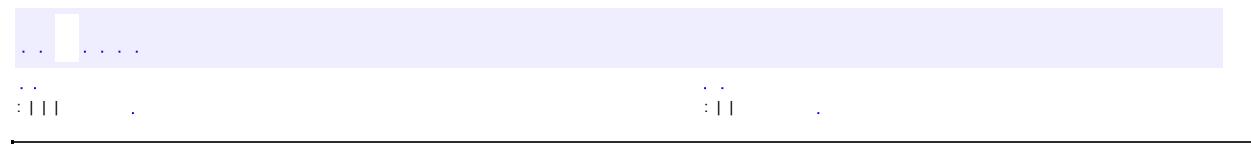


getDescription

[Description](#) **getDescription()**

:

a [Description](#) showing the tests to be run by the receiver





... 1 ...

... 1 ...



org.junit.runner Class Description

[java.lang.Object](#)
└ [org.junit.runner.Description](#)

public class

extends [Object](#)

A **Description** describes a test which is to be run or has been run. Descriptions can be atomic (a single test) or compound (containing children tests). Descriptions are used to provide feedback about the tests that are about to run (for example, the tree view visible in many IDEs) or tests that have been run (for example, the failures view).

Descriptions are implemented as a single class rather than a Composite because they are entirely informational. They contain no logic aside from counting their tests.

In the past, we used the raw **TestCases** and **TestSuites** to display the tree of tests. This was no longer viable in JUnit 4 because atomic tests no longer have a superclass below [Object](#). We needed a way to pass a class and name together. **Description** emerged from this.

:

[Request](#), [Runner](#)

static Description	EMPTY Describes a Runner which runs no tests
static Description	TEST_MECHANISM Describes a step in the test-running mechanism that goes so wrong no other description can be used (for example, an exception thrown from a Runner's constructor)

	<code>void addChild(Description description)</code> Add Description as a child of the receiver.
Description	<code>childlessCopy()</code>
static Description	<code>createSuiteDescription(Class<?> testClass)</code> Create a Description named after testClass
static Description	<code>createSuiteDescription(String name, Annotation... annotations)</code> Create a Description named name.
static Description	<code>createTestDescription(Class<?> clazz, String name)</code> Create a Description of a single test named name in the class clazz.
static Description	<code>createTestDescription(Class<?> clazz, String name, Annotation... annotations)</code> Create a Description of a single test named name in the class clazz.
boolean	<code>equals(Object obj)</code>
<T extends AnnotationT	

`getAnnotation\(Class<T> annotationType\)`

- `Collection<Annotation> getAnnotations\(\)`
- `ArrayList<Description> getChildren\(\)`
- `String getClassName\(\)`
- `String getDisplayName\(\)`
- `String getMethodName\(\)`
- `Class<?> getTestClass\(\)`
- `int hashCode\(\)`
- `boolean isEmpty\(\)`
- `boolean isSuite\(\)`
- `boolean isTest\(\)`
- `int testCount\(\)`
- `String toString\(\)`

[clone](#), [finalize](#), [getClass](#), [notify](#), [notifyAll](#), [wait](#), [wait](#), [wait](#)

EMPTY

public static final [Description](#) EMPTY

Describes a Runner which runs no tests

TEST_MECHANISM

public static final [Description](#) TEST_MECHANISM

Describes a step in the test-running mechanism that goes so wrong no other description can be used (for example, an exception thrown from a Runner's constructor)

createSuiteDescription

public static [Description](#) createSuiteDescription([String](#) name,
[Annotation](#)... annot

Create a Description named name. Generally, you will add children to this Description.

:

 name - the name of the Description

 annotations -

:

 a Description named name

createTestDescription

```
public static Description createTestDescription(Class<?> clazz,  
                          String name,  
                          Annotation... annota
```

Create a Description of a single test named name in the class clazz.
Generally, this will be a leaf Description.

:

 clazz - the class of the test
 name - the name of the test (a method name for test annotated with
 [Test](#))
 annotations - meta-data about the test, for downstream interpreters

:

 a Description named name

createTestDescription

```
public static Description createTestDescription(Class<?> clazz,  
                          String name)
```

Create a Description of a single test named name in the class clazz.
Generally, this will be a leaf Description. (This remains for binary
compatibility with clients of JUnit 4.3)

:

 clazz - the class of the test
 name - the name of the test (a method name for test annotated with
 [Test](#))
:

 a Description named name

createSuiteDescription

```
public static Description createSuiteDescription(Class<?> testClass)
```

Create a Description named after testClass

:

testClass - A [Class](#) containing tests
:
a Description of testClass

getDisplayName

public [String](#) **getDisplayName()**
:
a user-understandable label

addChild

public void **addChild**([Description](#) description)
Add Description as a child of the receiver.
:
description - the soon-to-be child.

getChildren

public [ArrayList<Description>](#) **getChildren()**
:
the receiver's children, if any

isSuite

public boolean **isSuite()**
:
true if the receiver is a suite

isTest

public boolean **isTest()**

:

 true if the receiver is an atomic test

testCount

public int **testCount()**

:

 the total number of atomic tests in the receiver

hashCode

public int **hashCode()**

:

[hashCode](#) in class [Object](#)

equals

public boolean **equals([Object](#) obj)**

:

[equals](#) in class [Object](#)

toString

public [String](#) **toString()**

:

[toString](#) in class [Object](#)

isEmpty

public boolean **isEmpty()**

:

true if this is a description of a Runner that runs no tests

childlessCopy

public [Description](#) **childlessCopy()**

:

a copy of this description, with no children (on the assumption that some of the children will be added back)

getAnnotation

public <T extends [Annotation](#)> T **getAnnotation([Class](#)<T> annotationTyp**

:

the annotation of type annotationType that is attached to this description node, or null if none exists

getAnnotations

public [Collection<Annotation>](#) **getAnnotations()**

:

all of the annotations attached to this description node

getTestClass

public [Class](#)<?> **getTestClass()**

:

If this describes a method invocation, the class of the test instance.

getClassName

```
public String getClassName()
```

:

If this describes a method invocation, the name of the class of the test instance

getMethodName

```
public String getMethodName()
```

:

If this describes a method invocation, the name of the method (or null if not)



The image shows a portion of a Java code editor interface. It displays two methods: `getClassName()` and `getMethodName()`. The code is written in a standard monospaced font. The methods are preceded by the `public` access modifier and followed by a pair of parentheses. The `String` return type is underlined in blue, indicating it is a hyperlink or a placeholder for a code template.



... 1 ...

... 1 ...



org.junit.runner Class JUnitCore

```
java.lang.Object
└ org.junit.runner.JUnitCore
```

```
public class JUnitCore
```

```
extends Object
```

JUnitCore is a facade for running tests. It supports running JUnit 4 tests, JUnit 3.8.x tests, and mixtures. To run tests from the command line, run `java org.junit.runner.JUnitCore TestClass1 TestClass2 ...`. For one-shot test runs, use the static method `runClasses(Class[])`. If you want to add special listeners, create an instance of `JUnitCore` first and use it to run the tests.

```
:
```

```
Result, RunListener, Request
```

JUnitCore()	
Create a new JUnitCore to run tests.	

void	addListener(RunListener listener) Add a listener to be notified as the tests run.
<u>String</u>	getVersion()
static void	main(String... args) Run the tests contained in the classes named in the args.
void	removeListener(RunListener listener) Remove a listener.
<u>Result</u>	run(Class<?>... classes) Run all the tests in classes.

<code>Result</code>	<code>run(Computer computer, Class<?>... classes)</code> Run all the tests in classes.
<code>Result</code>	<code>run(Request request)</code> Run all the tests contained in request.
<code>Result</code>	<code>run(Runner runner)</code> Do not use.
<code>Result</code>	<code>run(junit.framework.Test test)</code> Run all the tests contained in JUnit 3.8.x test.
<code>static Result</code>	<code>runClasses(Class<?>... classes)</code> Run the tests contained in classes.
<code>static Result</code>	<code>runClasses(Computer computer, Class<?>... classes)</code> Run the tests contained in classes.
<code>Result</code>	<code>runMain(org.junit.internal.JUnitSystem system, String... args)</code> Do not use.
<code>static void</code>	<code>runMainAndExit(org.junit.internal.JUnitSystem system, String... args)</code> Do not use.

java.lang. `Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`,
`toString`, `wait`, `wait`, `wait`

JUnitCore

`public JUnitCore()`

Create a new JUnitCore to run tests.

main

```
public static void main(String... args)
```

Run the tests contained in the classes named in the args. If all tests run successfully, exit with a status of 0. Otherwise exit with a status of 1. Write feedback while tests are running and write stack traces for all failed tests after the tests all complete.

:

args - names of classes in which to find tests to run

runMainAndExit

```
public static void runMainAndExit(org.junit.internal.JUnitSystem sys  
                                 String... args)
```

Do not use. Testing purposes only.

:

system -

runClasses

```
public static Result runClasses(Computer computer,  
                           Class<?>... classes)
```

Run the tests contained in classes. Write feedback while the tests are running and write stack traces for all failed tests after all tests complete. This is similar to [main\(String\[\]\)](#), but intended to be used programmatically.

:

computer - Helps construct Runners from classes

classes - Classes in which to find tests

:

a [Result](#) describing the details of the test run and the failed tests.

runClasses

```
public static Result runClasses(Class<?>... classes)
```

Run the tests contained in classes. Write feedback while the tests are running and write stack traces for all failed tests after all tests complete. This is similar to [main\(String\[\]\)](#), but intended to be used programmatically.

:

classes - Classes in which to find tests

:

a [Result](#) describing the details of the test run and the failed tests.

runMain

```
public Result runMain(org.junit.internal.JUnitSystem system,  
                      String... args)
```

Do not use. Testing purposes only.

:

system -

getVersion

```
public String getVersion()
```

:

the version number of this release

run

```
public Result run(Class<?>... classes)
```

Run all the tests in classes.

:

classes - the classes containing tests

:

a [Result](#) describing the details of the test run and the failed tests.

run

```
public Result run(Computer computer,  
                  Class<?>... classes)
```

Run all the tests in classes.

:

computer - Helps construct Runners from classes
classes - the classes containing tests

:

a [Result](#) describing the details of the test run and the failed tests.

run

```
public Result run(Request request)
```

Run all the tests contained in request.

:

request - the request describing tests

:

a [Result](#) describing the details of the test run and the failed tests.

run

```
public Result run(junit.framework.Test test)
```

Run all the tests contained in JUnit 3.8.x test. Here for backward compatibility.

:

test - the old-style test

:

a [Result](#) describing the details of the test run and the failed tests.

run

```
public Result run(Runner runner)
```

Do not use. Testing purposes only.

addListener

```
public void addListener(RunListener listener)
```

Add a listener to be notified as the tests run.

:

 listener - the listener to add

:

[RunListener](#)

removeListener

```
public void removeListener(RunListener listener)
```

Remove a listener.

:

 listener - the listener to remove



[org.junit.runner](#)

[Describable](#)

Àà

[Computer](#)

[Description](#)

[JUnitCore](#)

[Request](#)

[Result](#)

[Runner](#)

Annotation Types

[RunWith](#)



Package org.junit.runner

Provides classes used to describe, collect, run and analyze multiple tests.

:

Describable	Represents an object that can describe itself
--------------------	---

Computer	Represents a strategy for computing runners and suites.
Description	A Description describes a test which is to be run or has been run.
JUnitCore	JUnitCore is a facade for running tests.
Request	A Request is an abstract description of tests to be run.
Result	A Result collects and summarizes information from running multiple tests.
Runner	A Runner runs tests and notifies a RunNotifier of significant events as it does so.

Annotation Types Summary

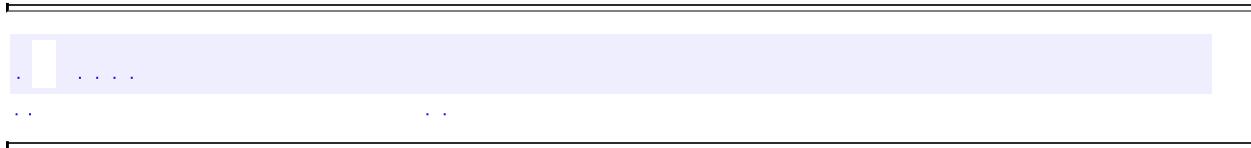
RunWith	When a class is annotated with @RunWith or extends a class annotated with @RunWith, JUnit will invoke the class it references to run the tests in that class instead of the runner built into JUnit.
----------------	--

Package org.junit.runner Description

Provides classes used to describe, collect, run and analyze multiple tests.

:

4.0





[PREV](#) [NEXT](#)



Hierarchy For Package org.junit.runner

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[Object](#)
 - org.junit.runner.[Computer](#)
 - org.junit.runner.
 - org.junit.runner.[JUnitCore](#)
 - org.junit.runner.[Request](#)
 - org.junit.runner.[Result](#)
 - org.junit.runner.[Runner](#) (implements org.junit.runner.[Describable](#))

Interface Hierarchy

- org.junit.runner.[**Describable**](#)

Annotation Type Hierarchy

- org.junit.runner.[RunWith](#) (implements java.lang.annotation.[Annotation](#))





... 1 ...

... 1 ...



org.junit.runner Class Request

[java.lang.Object](#)
└ [org.junit.runner.Request](#)

public abstract class **Request**

extends [Object](#)

A Request is an abstract description of tests to be run. Older versions of JUnit did not need such a concept--tests to be run were described either by classes containing tests or a tree of [Test](#)s. However, we want to support filtering and sorting, so we need a more abstract specification than the tests themselves and a richer specification than just the classes.

The flow when JUnit runs tests is that a Request specifies some tests to be run -> a [Runner](#) is created for each class implied by the Request -> the [Runner](#) returns a detailed [Description](#) which is a tree structure of the tests to be run.

Request()	
static Request	aClass(Class<?> clazz) Create a Request that, when processed, will run all the tests in a class.
static Request	classes(Class<?>... classes) Create a Request that, when processed, will run all the tests in a set of classes with the default Computer.
static Request	classes(Computer computer, Class<?>... classes) Create a Request that, when processed, will run all the tests in a set of classes.
	classWithoutSuiteMethod(Class<?> clazz)

<code>static Request</code>	Create a Request that, when processed, will run all the tests in a class.
<code>static Request</code>	<code>errorReport(Class<?> klass, Throwable cause)</code> Deprecated.
<code>Request</code>	<code>filterWith>Description desiredDescription)</code> Returns a Request that only runs contains tests whose <code>Description</code> equals <code>desiredDescription</code>
<code>Request</code>	<code>filterWith(Filter filter)</code> Returns a Request that only contains those tests that should run when filter is applied
<code>abstract Runner</code>	<code>getRunner()</code> Returns a <code>Runner</code> for this Request
<code>static Request</code>	<code>method(Class<?> clazz, String methodName)</code> Create a Request that, when processed, will run a single test.
<code>static Request</code>	<code>runner(Runner runner)</code>
<code>Request</code>	<code>sortWith(Comparator<Description> comparator)</code> Returns a Request whose Tests can be run in a certain order, defined by comparator For example, here is code to run a test suite in alphabetical order:

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

Request

`public Request()`

method

```
public static Request method(Class<?> clazz,  
                           String methodName)
```

Create a Request that, when processed, will run a single test. This is done by filtering out all other tests. This method is used to support rerunning single tests.

:

 clazz - the class of the test

 methodName - the name of the test

:

 a Request that will cause a single test be run

aClass

```
public static Request aClass(Class<?> clazz)
```

Create a Request that, when processed, will run all the tests in a class. The odd name is necessary because class is a reserved word.

:

 clazz - the class containing the tests

:

 a Request that will cause all tests in the class to be run

classWithoutSuiteMethod

```
public static Request classWithoutSuiteMethod(Class<?> clazz)
```

Create a Request that, when processed, will run all the tests in a class. If the class has a suite() method, it will be ignored.

:

 clazz - the class containing the tests

:

 a Request that will cause all tests in the class to be run

classes

```
public static Request classes(Computer computer,  
                 Class<?>... classes)
```

Create a Request that, when processed, will run all the tests in a set of classes.

:

 computer - Helps construct Runners from classes
 classes - the classes containing the tests

:

 a Request that will cause all tests in the classes to be run

classes

```
public static Request classes(Class<?>... classes)
```

Create a Request that, when processed, will run all the tests in a set of classes with the default Computer.

:

 classes - the classes containing the tests

:

 a Request that will cause all tests in the classes to be run

errorReport

[@Deprecated](#)

```
public static Request errorReport(Class<?> klass,  
                                 Throwable cause)
```

Deprecated.

Not used within JUnit. Clients should simply instantiate ErrorReportingRunner themselves

runner

```
public static Request runner(Runner runner)  
  
:  
    runner - the runner to return  
:  
    a Request that will run the given runner when invoked
```

getRunner

```
public abstract Runner getRunner()  
  
Returns a Runner for this Request  
  
:  
    corresponding Runner for this Request
```

filterWith

```
public Request filterWith(Filter filter)  
  
Returns a Request that only contains those tests that should run when  
filter is applied  
  
:  
    filter - The Filter to apply to this Request  
:  
    the filtered Request
```

filterWith

```
public Request filterWith(Description desiredDescription)  
  
Returns a Request that only runs contains tests whose Description equals  
desiredDescription
```

:
 desiredDescription - [Description](#) of those tests that should be run
:
the filtered Request

sortWith

public [Request](#) sortWith([Comparator<Description>](#) comparator)

Returns a Request whose Tests can be run in a certain order, defined by comparator For example, here is code to run a test suite in alphabetical order:

```
private static Comparator forward() {  
    return new Comparator() {  
        public int compare(Description o1, Descr  
                           return o1.getDisplayName().compa  
        }  
    };  
  
    public static main() {  
        new JUnitCore().run(Request.aClass(AllTests.clas  
    }
```

:
comparator - definition of the order of the tests in this Request
:
a Request with ordered Tests





... 1 ...

... 1 ...



org.junit.runner Class Result

[java.lang.Object](#)
└ [org.junit.runner.Result](#)

public class **Result**

extends [Object](#)

A Result collects and summarizes information from running multiple tests. Since tests are expected to run correctly, successful tests are only noted in the count of tests that ran.

[Result\(\)](#)

[RunListener](#)

[createListener\(\)](#)
Internal use only.

int

[getFailureCount\(\)](#)

[List<Failure>](#)

[getFailures\(\)](#)

int

[getIgnoreCount\(\)](#)

int

[getRunCount\(\)](#)

long

[getRunTime\(\)](#)

boolean

[wasSuccessful\(\)](#)

java.lang. Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)

Result

public Result()

getRunCount

public int getRunCount()

:

the number of tests run

getFailureCount

public int getFailureCount()

:

the number of tests that failed during the run

getRunTime

public long getRunTime()

:

the number of milliseconds it took to run the entire suite to run

getFailures

```
public List<Failure> getFailures()  
:  
    the Failures describing tests that failed and the problems they  
    encountered
```

getIgnoreCount

```
public int getIgnoreCount()  
:  
    the number of tests ignored during the run
```

wasSuccessful

```
public boolean wasSuccessful()  
:  
    true if all tests succeeded
```

createListener

```
public RunListener createListener()  
  
Internal use only.
```





... 1 ...

... 1 ...



org.junit.runner Class Runner

[java.lang.Object](#)
└ org.junit.runner.Runner

:

[Describable](#)

:

org.junit.internal.runners.JUnit38ClassRunner, [ParentRunner](#)

public abstract class **Runner**

extends [Object](#)

implements [Describable](#)

A Runner runs tests and notifies a [RunNotifier](#) of significant events as it does so. You will need to subclass Runner when using [RunWith](#) to invoke a custom runner. When creating a custom runner, in addition to implementing the abstract methods here you must also provide a constructor that takes as an argument the [Class](#) containing the tests.

The default runner implementation guarantees that the instances of the test case class will be constructed immediately before running the test and that the runner will retain no reference to the test case instances, generally making them available for garbage collection.

:

[Description](#), [RunWith](#)

[Runner\(\)](#)

||

	abstract getDescription() Description
abstract void	run(RunNotifier notifier) Run the tests for this runner.
int	testCount()

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)

Runner

public **Runner**()

:
[getDescription](#) in interface [Describable](#)
: a [Description](#) showing the tests to be run by the receiver

run

public abstract void **run**([RunNotifier](#) notifier)

Run the tests for this runner.

:

notifier - will be notified of events while tests are being run--tests
being started, finishing, and failing

testCount

`public int testCount()`

:

the number of tests to be run by the receiver



... [] ...

[] ...

CLASS

: [REQUIRED](#) | OPTIONAL

[ELEMENT](#)

org.junit.runner Annotation Type RunWith

```
@Retention(value=RUNTIME)
@Target(value=TYPE)
@Inherited
public @interface RunWith
```

When a class is annotated with @RunWith or extends a class annotated with @RunWith, JUnit will invoke the class it references to run the tests in that class instead of the runner built into JUnit. We added this feature late in development. While it seems powerful we expect the runner API to change as we learn how people really use it. Some of the classes that are currently internal will likely be refined and become public. For example, suites in JUnit 4 are built using RunWith, and a custom runner named Suite:

```
@RunWith(Suite.class)
@SuiteClasses(ATest.class, BTest.class, CTest.class)
public class ABCSuite { }
```

Required Element Summary

Class<? extends Runner>	value
---	-----------------------

value

```
public abstract Class<? extends Runner> value
```

:

a Runner class (must have a constructor that takes a single Class to run)

CLASS

: REQUIRED | OPTIONAL

ELEMENT



... 1.1.1

... 1.1.1



org.junit.runner.manipulation Class Filter

[java.lang.Object](#)
└ org.junit.runner.manipulation.Filter

public abstract class **Filter**

extends [Object](#)

The canonical case of filtering is when you want to run a single test method in a class. Rather than introduce runner API just for that one case, JUnit provides a general filtering mechanism. If you want to filter the tests to be run, extend **Filter** and apply an instance of your filter to the [Request](#) before running it (see [JUnitCore.run\(Request\)](#)). Alternatively, apply a **Filter** to a [Runner](#) before running tests (for example, in conjunction with [RunWith](#)).

static Filter	ALL A null Filter that passes all tests through.

	Filter()

void	apply(Object child) Invoke with a Runner to cause all tests it intends to run to first be checked with the filter.
abstract String	describe() Returns a textual description of this Filter
static Filter	matchMethodDescription(Description desiredDescription) Returns a Filter that only runs the single method described by desiredDescription

```
abstract shouldRun(Description description)  
boolean
```

java.lang. Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,  
toString, wait, wait, wait
```

ALL

```
public static Filter ALL
```

A null Filter that passes all tests through.

Filter

```
public Filter()
```

matchMethodDescription

```
public static Filter matchMethodDescription(Description desiredDescr
```

Returns a Filter that only runs the single method described by
desiredDescription

shouldRun

```
public abstract boolean shouldRun(Description description)
```

:

description - the description of the test to be run
:
true if the test should be run

describe

```
public abstract String describe()
```

Returns a textual description of this Filter

:
a textual description of this Filter

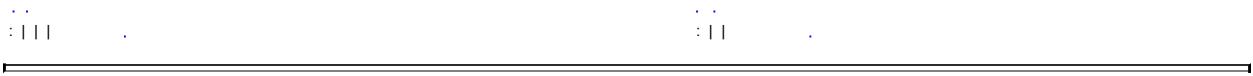
apply

```
public void apply(Object child)  
throws NoTestsRemainException
```

Invoke with a [Runner](#) to cause all tests it intends to run to first be checked with the filter. Only those that pass the filter will be run.

:
child - the runner to be filtered by the receiver
:
[NoTestsRemainException](#) - if the receiver removes all tests





org.junit.runner.manipulation Interface Filterable

All Known Implementing Classes:

[AllTests](#), [BlockJUnit4ClassRunner](#),
[org.junit.internal.runners.JUnit38ClassRunner](#), [JUnit4](#), [Parameterized](#),
[ParentRunner](#), [Suite](#), [org.junit.internal.runners.SuiteMethod](#)

```
public interface Filterable
```

Runners that allow filtering should implement this interface. Implement [filter\(Filter\)](#) to remove tests that don't pass the filter.

void	filter(Filter filter)
	Remove tests that don't pass the parameter filter.

filter

```
void filter(Filter filter\)
           throws NoTestsRemainException
```

Remove tests that don't pass the parameter filter.

:

 filter - the [Filter](#) to apply

:

[NoTestsRemainException](#) - if all tests are filtered out





... 1 ...

... 1 ...



org.junit.runner.manipulation Class NoTestsRemainException

```
java.lang.Object
└ java.lang.Throwable
  └ java.lang.Exception
    └ org.junit.runner.manipulation.NoTestsRemainException

:
```

[Serializable](#)

```
public class NoTestsRemainException
extends Exception
```

Thrown when a filter removes all tests from a runner.

```
:
```

[Serialized Form](#)

[NoTestsRemainException\(\)](#)

java.lang. [Throwable](#)

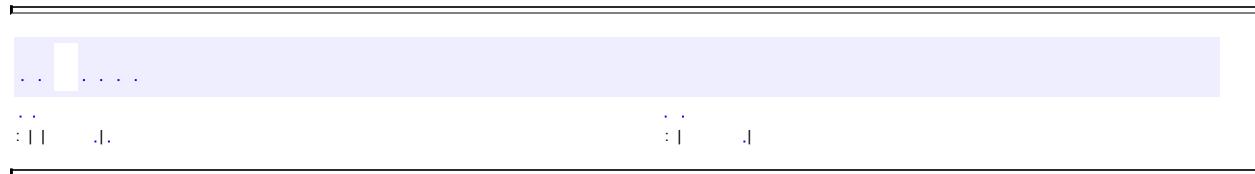
[fillInStackTrace](#), [getCause](#), [getLocalizedMessage](#), [getMessage](#),
[getStackTrace](#), [initCause](#), [printStackTrace](#), [printStackTrace](#),
[printStackTrace](#), [setStackTrace](#), [toString](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

NoTestsRemainException

```
public NoTestsRemainException()
```



[org.junit.runner.manipulation](#)

[Filterable](#)

[Sortable](#)

[Filter](#)

[Sorter](#)

[NoTestsRemainException](#)



Package org.junit.runner.manipulation

Provides classes to [filter](#) or [sort](#) tests.

:

Filterable	Runners that allow filtering should implement this interface.
Sortable	Interface for runners that allow sorting of tests.

Filter	The canonical case of filtering is when you want to run a single test method in a class.
Sorter	A Sorter orders tests.

Exception Summary

NoTestsRemainException	Thrown when a filter removes all tests from a runner.
--	---

Package org.junit.runner.manipulation Description

Provides classes to [filter](#) or [sort](#) tests.

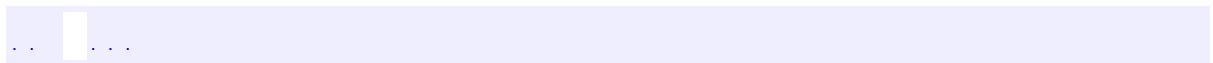
:

4.0

:

[Runner](#)





[PREV](#) [NEXT](#)



Hierarchy For Package org.junit.runner.manipulation

Package Hierarchies:

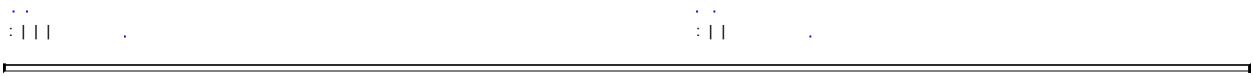
[All Packages](#)

Class Hierarchy

- java.lang.[Object](#)
 - org.junit.runner.manipulation.[Filter](#)
 - org.junit.runner.manipulation.[Sorter](#) (implements
java.util.[Comparator](#)<T>)
 - java.lang.[Throwable](#) (implements java.io.[Serializable](#))
 - java.lang.[.](#)
 - org.junit.runner.manipulation.[NoTestsRemainException](#)

Interface Hierarchy

- org.junit.runner.manipulation.[Filterable](#)
 - org.junit.runner.manipulation.[Sortable](#)
-
- 
- [PREV](#) [NEXT](#)
-



org.junit.runner.manipulation Interface Sortable

All Known Implementing Classes:

[AllTests](#), [BlockJUnit4ClassRunner](#),
[org.junit.internal.runners.JUnit38ClassRunner](#), [JUnit4](#), [Parameterized](#),
[ParentRunner](#), [Suite](#), [org.junit.internal.runners.SuiteMethod](#)

```
public interface Sortable
```

Interface for runners that allow sorting of tests. By sorting tests based on when they last failed, most recently failed first, you can reduce the average time to the first test failing. Test sorting should not be used to cope with order dependencies between tests. Tests that are isolated from each other are less expensive to maintain and can be run individually.

void	sort (Sorter sorter) Sorts the tests using sorter

sort

```
void sort(Sorter sorter)
```

Sorts the tests using sorter

:

sorter - the [Sorter](#) to use for sorting the tests

...
...	...



CLASS

:| .JL.

CLASS

:| .JL.

org.junit.runner.manipulation Class Sorter

```
java.lang.Object
└ org.junit.runner.manipulation.Sorter

:

Comparator<Description>
```

```
public class Sorter
extends Object
implements Comparator<Description>
```

A Sorter orders tests. In general you will not need to use a Sorter directly. Instead, use [Request.sortWith\(Comparator\)](#).

static Sorter	NULL
	NULL is a Sorter that leaves elements in an undefined order

Sorter (Comparator<Description> comparator)
Creates a Sorter that uses comparator to sort tests

void	apply(Object object)
	Sorts the test in runner using comparator
int	compare (Description o1, Description o2)

java.lang.	Object
	clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

Methods inherited from interface java.util.Comparator

[equals](#)

NULL

public static [Sorter](#) NULL

NULL is a Sorter that leaves elements in an undefined order

Sorter

public [Sorter](#)([Comparator](#)<[Description](#)> comparator)

Creates a Sorter that uses comparator to sort tests

:

comparator - the [Comparator](#) to use when sorting tests

apply

public void apply([Object](#) object)

Sorts the test in runner using comparator

:

object -

compare

```
public int compare(Description o1,  
                  Description o2)  
  
:  
    compare in interface Comparator<Description>
```



The screenshot shows a portion of a Java code editor. A class definition is visible, starting with 'CLASS' and ending with a closing brace '}'. The code is displayed in a light blue background with dark blue syntax highlighting for keywords like 'CLASS' and '}'.



... 1 2 ...

... 1 2 ...



org.junit.runner.notification Class Failure

[java.lang.Object](#)
└ [org.junit.runner.notification.Failure](#)

public class **Failure**

extends [Object](#)

A Failure holds a description of the failed test and the exception that was thrown while running it. In most cases the [Description](#) will be of a single test. However, if problems are encountered while constructing the test (for example, if a [BeforeClass](#) method is not static), it may describe something other than a single test.

Failure (Description description, Throwable thrownException) Constructs a Failure with the given description and exception.	

Description	getDescription()
Throwable	getException()
String	getMessage() Convenience method
String	getTestHeader()
String	getTrace() Convenience method
String	toString()

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[wait](#), [wait](#), [wait](#)

Failure

**public Failure([Description](#) description,
 [Throwable](#) thrownException)**

Constructs a Failure with the given description and exception.

:

 description - a [Description](#) of the test that failed
 thrownException - the exception that was thrown while running the
 test

getTestHeader

public [String](#) getTestHeader()

:

 a user-understandable label for the test

getDescription

public [Description](#) getDescription()

:

 the raw description of the context of the failure.

getException

```
public Throwable getException()
```

:

the exception thrown

toString

```
public String toString()
```

:

[toString](#) in class [Object](#)

getTrace

```
public String getTrace()
```

Convenience method

:

the printed form of the exception

getMessage

```
public String getMessage()
```

Convenience method

:

the message of the thrown exception



... [REDACTED] ...

:| :| :| :| :|

[org.junit.runner.notification](#)

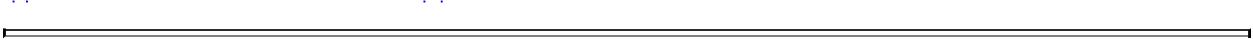
↳ [Failure](#)

[RunListener](#)

[RunNotifier](#)

↳
↳ [Exception](#)

[StoppedByUserException](#)



Package org.junit.runner.notification

Provides information about a test run.

:

<u>Failure</u>	A Failure holds a description of the failed test and the exception that was thrown while running it.
<u>RunListener</u>	If you need to respond to the events during a test run, extend RunListener and override the appropriate methods.
<u>RunNotifier</u>	If you write custom runners, you may need to notify JUnit of your progress running tests.

Exception Summary

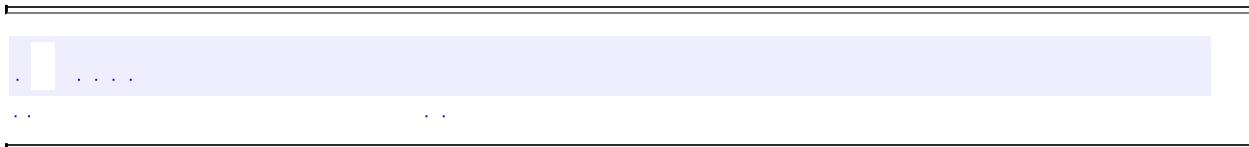
<u>StoppedByUserException</u>	Thrown when a user has requested that the test run stop.
---	--

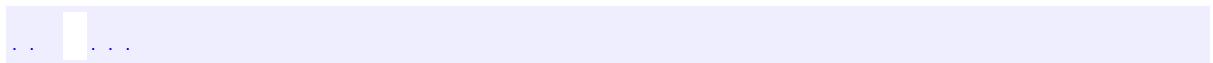
Package org.junit.runner.notification Description

Provides information about a test run.

:

4.0





[PREV](#) [NEXT](#)



Hierarchy For Package org.junit.runner.notification

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[**Object**](#)
 - org.junit.runner.notification.[**Failure**](#)
 - org.junit.runner.notification.[**RunListener**](#)
 - org.junit.runner.notification.[**RunNotifier**](#)
 - java.lang.[**Throwable**](#) (implements java.io.[**Serializable**](#))
 - java.lang.[**RuntimeException**](#)
 - org.junit.runner.notification.[**StoppedByUserException**](#)



[PREV](#) [NEXT](#)



... 1 ...

... 1 ...



org.junit.runner.notification Class RunListener

```
java.lang.Object
└ org.junit.runner.notification.RunListener
```

```
public class RunListener
extends Object
```

If you need to respond to the events during a test run, extend `RunListener` and override the appropriate methods. If a listener throws an exception while processing a test event, it will be removed for the remainder of the test run.

For example, suppose you have a `Cowbell` class that you want to make a noise whenever a test fails. You could write:

```
public class RingingListener extends RunListener {
    public void testFailure(Failure failure) {
        Cowbell.ring();
    }
}
```

To invoke your listener, you need to run your tests through `JUnitCore`.

```
public void main(String... args) {
    JUnitCore core= new JUnitCore();
    core.addListener(new RingingListener());
    core.run(MyTestClass.class);
}

:
```

[JUnitCore](#)

[RunListener\(\)](#)

void	testAssumptionFailure(Failure failure) Called when an atomic test flags that it assumes a condition that is false
void	testFailure(Failure failure) Called when an atomic test fails.
void	testFinished(Description description) Called when an atomic test has finished, whether the test succeeds or fails.
void	testIgnored(Description description) Called when a test will not be run, generally because a test method is annotated with Ignore .
void	testRunFinished(Result result) Called when all tests have finished
void	testRunStarted(Description description) Called before any tests have been run.
void	testStarted(Description description) Called when an atomic test is about to be started.

java.lang. [**Object**](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

RunListener

public **RunListener()**

testRunStarted

public void **testRunStarted(Description description)**

```
throws Exception
```

Called before any tests have been run.

```
:  
    description - describes the tests to be run  
:  
Exception
```

testRunFinished

```
public void testRunFinished(Result result)  
    throws Exception
```

Called when all tests have finished

```
:  
    result - the summary of the test run, including all the tests that failed  
:  
Exception
```

testStarted

```
public void testStarted(Description description)  
    throws Exception
```

Called when an atomic test is about to be started.

```
:  
    description - the description of the test that is about to be run  
    (generally a class and method name)  
:  
Exception
```

testFinished

```
public void testFinished(Description description)  
    throws Exception
```

Called when an atomic test has finished, whether the test succeeds or fails.

:

- description - the description of the test that just ran

:

- [Exception](#)

testFailure

```
public void testFailure(Failure failure)
    throws Exception
```

Called when an atomic test fails.

:

- failure - describes the test that failed and the exception that was thrown

:

- [Exception](#)

testAssumptionFailure

```
public void testAssumptionFailure(Failure failure)
```

Called when an atomic test flags that it assumes a condition that is false

:

- failure - describes the test that failed and the AssumptionViolatedException that was thrown

testIgnored

```
public void testIgnored(Description description)
    throws Exception
```

Called when a test will not be run, generally because a test method is annotated with [Ignore](#).

- :
 - description - describes the test that will not be run
 - :
 - [Exception](#)
-





... 1 ...

... 1 ...



org.junit.runner.notification Class RunNotifier

[java.lang.Object](#)
└ org.junit.runner.notification.RunNotifier

public class **RunNotifier**

extends [Object](#)

If you write custom runners, you may need to notify JUnit of your progress running tests. Do this by invoking the `RunNotifier` passed to your implementation of [Runner.run\(RunNotifier\)](#). Future evolution of this class is likely to move [fireTestRunStarted\(Description\)](#) and [fireTestRunFinished\(Result\)](#) to a separate class since they should only be called once per run.

[RunNotifier\(\)](#)

void	addFirstListener(RunListener listener) Internal use only.
void	addListener(RunListener listener) Internal use only
void	fireTestAssumptionFailed(Failure failure) Invoke to tell listeners that an atomic test flagged that it assumed something false.
void	fireTestFailure(Failure failure) Invoke to tell listeners that an atomic test failed.
void	fireTestFinished(Description description) Invoke to tell listeners that an atomic test finished.
void	fireTestIgnored(Description description)

	Invoke to tell listeners that an atomic test was ignored.
void	fireTestRunFinished(Result result) Do not invoke.
void	fireTestRunStarted(Description description) Do not invoke.
void	fireTestStarted(Description description) Invoke to tell listeners that an atomic test is about to start.
void	pleaseStop() Ask that the tests run stop before starting the next test.
void	removeListener(RunListener listener) Internal use only

java.lang. [**Object**](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)

RunNotifier

public **RunNotifier()**

--

addListener

public void **addListener**([RunListener](#) listener)

Internal use only

--

removeListener

public void **removeListener**([RunListener](#) listener)

Internal use only

fireTestRunStarted

```
public void fireTestRunStarted(Description description)
```

Do not invoke.

fireTestRunFinished

```
public void fireTestRunFinished(Result result)
```

Do not invoke.

fireTestStarted

```
public void fireTestStarted(Description description)
    throws StoppedByUserException
```

Invoke to tell listeners that an atomic test is about to start.

:

 description - the description of the atomic test (generally a class and method name)

:

[StoppedByUserException](#) - thrown if a user has requested that the test run stop

fireTestFailure

```
public void fireTestFailure(Failure failure)
```

Invoke to tell listeners that an atomic test failed.

:

 failure - the description of the test that failed and the exception

thrown

fireTestAssumptionFailed

```
public void fireTestAssumptionFailed(Failure failure)
```

Invoke to tell listeners that an atomic test flagged that it assumed something false.

:

failure - the description of the test that failed and the AssumptionViolatedException thrown

fireTestIgnored

```
public void fireTestIgnored(Description description)
```

Invoke to tell listeners that an atomic test was ignored.

:

description - the description of the ignored test

fireTestFinished

```
public void fireTestFinished(Description description)
```

Invoke to tell listeners that an atomic test finished. Always invoke [fireTestFinished](#)([Description](#)) if you invoke [fireTestStarted](#)([Description](#)) as listeners are likely to expect them to come in pairs.

:

description - the description of the test that finished

pleaseStop

```
public void pleaseStop()
```

Ask that the tests run stop before starting the next test. Phrased politely because the test currently running will not be interrupted. It seems a little odd to put this functionality here, but the `RunNotifier` is the only object guaranteed to be shared amongst the many runners involved.

addFirstListener

```
public void addFirstListener(RunListener listener)
```

Internal use only. The Result's listener must be first.

A screenshot of a Java code editor showing the implementation of the `addFirstListener` method. The code is identical to the one shown above, but it is part of a larger context. The editor has a light blue background with dark blue syntax highlighting. The method signature is at the top, followed by three horizontal ellipses (...). Below that, there are two pairs of vertical ellipses (....) indicating continuation of the code. The code ends with a closing brace } and a final pair of vertical ellipses



CLASS

:| | :| |

:| | :| |



org.junit.runner.notification Class StoppedByUserException

```
java.lang.Object
└ java.lang.Throwable
  └ java.lang.Exception
    └ java.lang.RuntimeException
      └ org.junit.runner.notification.StoppedByUserException

:

Serializable
```

```
public class StoppedByUserException
extends RuntimeException
```

Thrown when a user has requested that the test run stop. Writers of test running GUIs should be prepared to catch a `StoppedByUserException`.

```
:

RunNotifier, Serialized Form
```

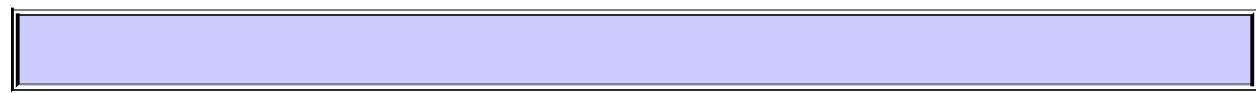
```
StoppedByUserException()
```

```
java.lang. Throwable
```

```
fillInStackTrace, getCause, getLocalizedMessage, getMessage,
getStackTrace, initCause, printStackTrace, printStackTrace,
printStackTrace, setStackTrace, toString
```

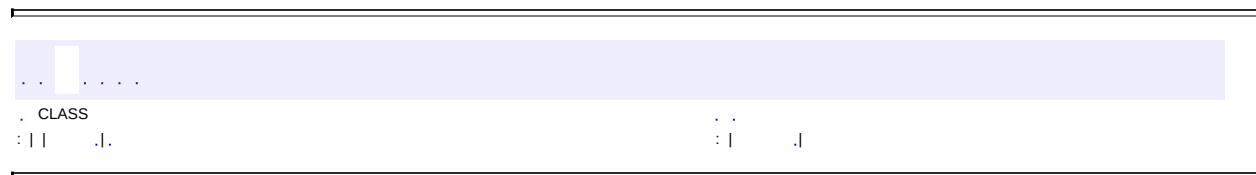
```
java.lang. Object
```

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
wait, wait, wait
```



StoppedByUserException

```
public StoppedByUserException()
```





... 1 2 ...

... 1 2 ...



org.junit.runners Class AllTests

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.internal.runners.JUnit38ClassRunner
        └ org.junit.internal.runners.SuiteMethod
            └ org.junit.runners.AllTests

:
Describable, Filterable, Sortable
```

```
public class AllTests
extends org.junit.internal.runners.SuiteMethod
```

Runner for use with JUnit 3.8.x-style AllTests classes (those that only implement a static suite() method). For example:

```
@RunWith(AllTests.class)
public class ProductTests {
    public static junit.framework.Test suite() {
        ...
    }
}
```

AllTests(Class<?> klass)
Only called reflectively.

org.junit.internal.runners.SuiteMethod
testFromSuiteMethod

org.junit.internal.runners.JUnit38ClassRunner

```
createAdaptingListener, filter, getDescription, run, sort
```

org.junit.runner.	Runner
--------------------------	-------------------------------

testCount

java.lang.	Object
-------------------	-------------------------------

clone , equals , finalize , getClass , hashCode , notify , notifyAll , toString , wait , wait , wait

AllTests

```
public AllTests(Class<?> klass)
    throws Throwable
```

Only called reflectively. Do not use programmatically.

:

[Throwable](#)

... | ...

:| :| .| .|

:| :| .|



... 1 ...

... 1 ...



org.junit.runners Class BlockJUnit4ClassRunner

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.runners.ParentRunner<org.junit.runners.model.FrameworkMethod>
        └ org.junit.runners.BlockJUnit4ClassRunner
```

:

[Describable](#), [Filterable](#), [Sortable](#)

:

[JUnit4](#)

```
public class BlockJUnit4ClassRunner
extends ParentRunner<org.junit.runners.model.FrameworkMethod>
```

Implements the JUnit 4 standard test case class model, as defined by the annotations in the org.junit package. Many users will never notice this class: it is now the default test class runner, but it should have exactly the same behavior as the old test class runner ([JUnit4ClassRunner](#)). [BlockJUnit4ClassRunner](#) has advantages for writers of custom JUnit runners that are slight changes to the default behavior, however:

- It has a much simpler implementation based on [Statements](#), allowing new operations to be inserted into the appropriate point in the execution flow.
 - It is published, and extension and reuse are encouraged, whereas [JUnit4ClassRunner](#) was in an internal package, and is now deprecated.
-

BlockJUnit4ClassRunner(Class <?> klass)
Creates a BlockJUnit4ClassRunner to run klass

collectInitializationErrors (List < InitializationError > errors)
--

<code>protected void</code>	Adds to errors a throwable for from ParentRunner.getTestClass()
<code>protected List<org.junit.runners.model.FrameworkMethod></code>	computeTestMethods() Returns the methods that run tes
<code>protected Object</code>	createTest() Returns a new fixture for runnin
<code>protected Description</code>	describeChild(org.junit.runners. Returns a Description for chil list returned by ParentRunner.getChildren()
<code>protected List<org.junit.runners.model.FrameworkMethod></code>	getChildren() Returns a list of objects that defi
<code>protected org.junit.runners.model.Statement</code>	methodBlock(org.junit.runners.mo Returns a Statement that, when e or throws an exception if method fails.
<code>protected org.junit.runners.model.Statement</code>	methodInvoker(org.junit.runners. Returns a Statement that invoke
<code>protected org.junit.runners.model.Statement</code>	possiblyExpectingExceptions(org. Object test, org.junit.runners.m Deprecated. Will be private soon
<code>protected List<org.junit.rules.MethodRule></code>	rules(Object test)
<code>protected void</code>	runChild(org.junit.runners.model RunNotifier notifier) Runs the test corresponding to c the list returned by ParentRunner.get
<code>protected String</code>	testName(org.junit.runners.model Returns the name that describes
<code>protected void</code>	validateConstructor(List<Throwab Adds to errors if the test class l takes parameters.
<code>protected void</code>	validateInstanceMethods(List<Thr Deprecated. unused API, will ge
<code>protected void</code>	validateOnlyOneConstructor(List< Adds to errors if the test class l
<code>protected void</code>	validateTestMethods(List<Throwab

	Adds to errors for each method instance method with no arguments.
protected void validateZeroArgConstructor(List<Method> methods)	Adds to errors if the test class's override)
protected org.junit.runners.model.Statement withAfters(org.junit.runners.model.Statement after)	Deprecated. Will be private soon
protected org.junit.runners.model.Statement withBefores(org.junit.runners.model.Statement before)	Deprecated. Will be private soon
protected org.junit.runners.model.Statement withPotentialTimeout(org.junit.runner.RunWith annotation, Object test, org.junit.runners.model.Statement statement)	Deprecated. Will be private soon

org.junit.runners. [ParentRunner](#)

[childrenInvoker](#), [classBlock](#), [filter](#), [getDescription](#), [getName](#),
[getTestClass](#), [run](#), [setScheduler](#), [sort](#),
[validatePublicVoidNoArgMethods](#), [withAfterClasses](#),
[withBeforeClasses](#)

org.junit.runner. [Runner](#)

[testCount](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)

BlockJUnit4ClassRunner

```
public BlockJUnit4ClassRunner(Class<?> klass)
    throws org.junit.runners.model.InitializationError
```

Creates a BlockJUnit4ClassRunner to run klass

:

org.junit.runners.model.InitializationError - if the test class is malformed.

runChild

```
protected void runChild(org.junit.runners.model.FrameworkMethod meth  
RunNotifier notifier)
```

Description copied from class: [ParentRunner](#)

Runs the test corresponding to child, which can be assumed to be an element of the list returned by [ParentRunner.getChildren\(\)](#). Subclasses are responsible for making sure that relevant test events are reported through notifier

:

[runChild](#) in class
[ParentRunner<org.junit.runners.model.FrameworkMethod>](#)

describeChild

```
protected Description describeChild(org.junit.runners.model.FrameworkMethod)
```

Description copied from class: [ParentRunner](#)

Returns a [Description](#) for child, which can be assumed to be an element of the list returned by [ParentRunner.getChildren\(\)](#)

:

[describeChild](#) in class
[ParentRunner<org.junit.runners.model.FrameworkMethod>](#)

getChildren

```
protected List<org.junit.runners.model.FrameworkMethod> getChildren()
```

Description copied from class: [ParentRunner](#)

Returns a list of objects that define the children of this Runner.

:

[getChildren](#) in class
[ParentRunner<org.junit.runners.model.FrameworkMethod>](#)

computeTestMethods

protected [List<org.junit.runners.model.FrameworkMethod>](#) **computeTestM**

Returns the methods that run tests. Default implementation returns all methods annotated with @Test on this class and superclasses that are not overridden.

collectInitializationErrors

protected void **collectInitializationErrors**([List<Throwable>](#) errors)

Description copied from class: [ParentRunner](#)

Adds to errors a throwable for each problem noted with the test class (available from [ParentRunner.getTestClass\(\)](#)). Default implementation adds an error for each method annotated with @BeforeClass or @AfterClass that is not public static void with no arguments.

:

[collectInitializationErrors](#) in class
[ParentRunner<org.junit.runners.model.FrameworkMethod>](#)

validateConstructor

protected void **validateConstructor**([List<Throwable>](#) errors)

Adds to errors if the test class has more than one constructor, or if the constructor takes parameters. Override if a subclass requires different validation rules.

validateOnlyOneConstructor

```
protected void validateOnlyOneConstructor(List<Throwable> errors)
```

Adds to errors if the test class has more than one constructor (do not override)

validateZeroArgConstructor

```
protected void validateZeroArgConstructor(List<Throwable> errors)
```

Adds to errors if the test class's single constructor takes parameters (do not override)

validateInstanceMethods

@Deprecated

```
protected void validateInstanceMethods(List<Throwable> errors)
```

Deprecated. *unused API, will go away in future version*

Adds to errors for each method annotated with @Test, @Before, or @After that is not a public, void instance method with no arguments.

validateTestMethods

```
protected void validateTestMethods(List<Throwable> errors)
```

Adds to errors for each method annotated with @Test that is not a public, void instance method with no arguments.

createTest

```
protected Object createTest()
    throws Exception
```

Returns a new fixture for running a test. Default implementation executes the test class's no-argument constructor (validation should have ensured one exists).

:

[Exception](#)

testName

protected [String](#) **testName**(org.junit.runners.model.FrameworkMethod me

Returns the name that describes method for [Descriptions](#). Default implementation is the method's name

methodBlock

protected org.junit.runners.model.Statement **methodBlock**(org.junit.ru

Returns a Statement that, when executed, either returns normally if method passes, or throws an exception if method fails. Here is an outline of the default implementation:

- Invoke method on the result of `createTest()`, and throw any exceptions thrown by either operation.
- HOWEVER, if method's `@Test` annotation has the `expecting` attribute, return normally only if the previous step threw an exception of the correct type, and throw an exception otherwise.
- HOWEVER, if method's `@Test` annotation has the `timeout` attribute, throw an exception if the previous step takes more than the specified number of milliseconds.
- ALWAYS allow `@Rule` fields to modify the execution of the above steps. A Rule may prevent all execution of the above steps, or add additional behavior before and after, or modify thrown exceptions. For more information, see `MethodRule`
- ALWAYS run all non-overridden `@Before` methods on this class and superclasses before any of the previous steps; if any throws an Exception, stop execution and pass the exception on.
- ALWAYS run all non-overridden `@After` methods on this class and

superclasses after any of the previous steps; all After methods are always executed: exceptions thrown by previous steps are combined, if necessary, with exceptions from After methods into a `MultipleFailureException`.

This can be overridden in subclasses, either by overriding this method, or the implementations creating each sub-statement.

methodInvoker

```
protected org.junit.runners.model.Statement methodInvoker(org.junit.  
Object tes
```

Returns a Statement that invokes method on test

possiblyExpectingExceptions

@Deprecated

```
protected org.junit.runners.model.Statement possiblyExpectingExcepti
```

Deprecated. *Will be private soon: use Rules instead*

Returns a Statement: if method's @Test annotation has the expecting attribute, return normally only if next throws an exception of the correct type, and throw an exception otherwise.

withPotentialTimeout

@Deprecated

```
protected org.junit.runners.model.Statement withPotentialTimeout(org
```

Deprecated. *Will be private soon: use Rules instead*

Returns a Statement: if method's @Test annotation has the timeout attribute, throw an exception if next takes more than the specified number

of milliseconds.

withBefores

@Deprecated
protected org.junit.runners.model.Statement **withBefores**(org.junit.ru
o
o

Deprecated. *Will be private soon: use Rules instead*

Returns a Statement: run all non-overridden @Before methods on this class and superclasses before running next; if any throws an Exception, stop execution and pass the exception on.

withAfters

@Deprecated
protected org.junit.runners.model.Statement **withAfters**(org.junit.run
ob
or

Deprecated. *Will be private soon: use Rules instead*

Returns a Statement: run all non-overridden @After methods on this class and superclasses before running next; all After methods are always executed: exceptions thrown by previous steps are combined, if necessary, with exceptions from After methods into a MultipleFailureException.

rules

protected [List](#)<org.junit.rules.MethodRule> **rules**([Object](#) test)

:

the MethodRules that can transform the block that runs each method in the tested class.



... [View](#)

11. [View](#)

11. [View](#)



... 1 2 ...

... 1 2 ...



org.junit.runners Class JUnit4

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.runners.ParentRunner<org.junit.runners.model.FrameworkMethod>
        └ org.junit.runners.BlockJUnit4ClassRunner
            └ org.junit.runners.JUnit4

:

Describable, Filterable, Sortable
```

```
public final class JUnit4
extends BlockJUnit4ClassRunner
```

Aliases the current default JUnit 4 class runner, for future-proofing. If future versions of JUnit change the default Runner class, they will also change the definition of this class. Developers wanting to explicitly tag a class as a JUnit 4 class should use `@RunWith(JUnit4.class)`, not, for example in JUnit 4.5, `@RunWith(BlockJUnit4ClassRunner.class)`. This is the only way this class should be used--any extension that depends on the implementation details of this class is likely to break in future versions.

JUnit4(Class<?> klass) Constructs a new instance of the default runner
--

org.junit.runners. BlockJUnit4ClassRunner
collectInitializationErrors, computeTestMethods, createTest, describeChild, getChildren, methodBlock, methodInvoker, possiblyExpectingExceptions, rules, runChild, testName, validateConstructor, validateInstanceMethods, validateOnlyOneConstructor, validateTestMethods, validateZeroArgConstructor, withAfters, withBefores,

[withPotentialTimeout](#)

org.junit.runners. [ParentRunner](#)

[childrenInvoker](#), [classBlock](#), [filter](#), [getDescription](#), [getName](#),
[getTestClass](#), [run](#), [setScheduler](#), [sort](#),
[validatePublicVoidNoArgMethods](#), [withAfterClasses](#),
[withBeforeClasses](#)

org.junit.runner. [Runner](#)

[testCount](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#), [wait](#)

JUnit4

```
public JUnit4(Class<?> klass)
    throws org.junit.runners.model.InitializationError
```

Constructs a new instance of the default runner

:

 org.junit.runners.model.InitializationError

...

..

..

[org.junit.runners](#)

Àà [AllTests](#)

[BlockJUnit4ClassRunner](#)

[JUnit4](#)

[Parameterized](#)

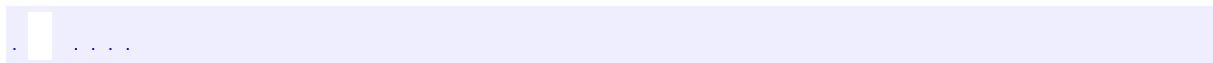
[ParentRunner](#)

[Suite](#)

Annotation Types

[Parameterized.Parameters](#)

[Suite.SuiteClasses](#)



PACKAGE



Package org.junit.runners

Provides standard [Runner](#) implementations.

:

AllTests	Runner for use with JUnit 3.8.x-style AllTests classes (those that only implement a static suite() method).
BlockJUnit4ClassRunner	Implements the JUnit 4 standard test case class model, as defined by the annotations in the org.junit package.
JUnit4	Aliases the current default JUnit 4 class runner, for future-proofing.
Parameterized	The custom runner Parameterized implements parameterized tests.
ParentRunner<T>	Provides most of the functionality specific to a Runner that implements a "parent node" in the test tree, with children defined by objects of some data type T.
Suite	Using Suite as a runner allows you to manually build a suite containing tests from many classes.

Annotation Types Summary

Parameterized.Parameters	Annotation for a method which provides parameters to be injected into the test class constructor by Parameterized
Suite.SuiteClasses	The SuiteClasses annotation specifies the classes to be run when a class annotated with @RunWith(Suite.class) is run.

Package org.junit.runners Description

Provides standard [Runner](#) implementations.

:

4.0

:

[Runner](#), [BlockJUnit4ClassRunner](#)





[PREV](#)



Hierarchy For Package org.junit.runners

Package Hierarchies:

[All Packages](#)

Class Hierarchy

- java.lang.[Object](#)
 - org.junit.runner.[Runner](#) (implements org.junit.runner.[Describable](#))
 - org.junit.internal.runners.JUnit38ClassRunner (implements org.junit.runner.manipulation.[Filterable](#), org.junit.runner.manipulation.[Sortable](#))
 - org.junit.internal.runners.SuiteMethod
 - org.junit.runners.[AllTests](#)
 - org.junit.runners.[ParentRunner](#)<T> (implements org.junit.runner.manipulation.[Filterable](#), org.junit.runner.manipulation.[Sortable](#))
 - org.junit.runners.[BlockJUnit4ClassRunner](#)
 - org.junit.runners.[JUnit4](#)
 - org.junit.runners.[Suite](#)
 - org.junit.runners.[Parameterized](#)

Annotation Type Hierarchy

- org.junit.runners.[Suite.SuiteClasses](#) (implements
java.lang.annotation.[Annotation](#))
 - org.junit.runners.[Parameterized.Parameters](#) (implements
java.lang.annotation.[Annotation](#))
-
- 
- ... PREV ...
-



... [NESTED](#) || .|.

... | .|.



org.junit.runners Class Parameterized

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.runners.ParentRunner<Runner>
        └ org.junit.runners.Suite
            └ org.junit.runners.Parameterized

:

Describable, Filterable, Sortable
```

```
public class Parameterized
```

```
extends Suite
```

The custom runner Parameterized implements parameterized tests. When running a parameterized test class, instances are created for the cross-product of the test methods and the test data elements.

For example, to test a Fibonacci function, write:

```
@RunWith(Parameterized.class)
public class FibonacciTest {
    @Parameters
    public static List<Object[]> data() {
        return Arrays.asList(new Object[][] {
            {Fibonacci,
             { { 0, 0 }, { 1, 1 }, { 2, 1 }, { 3, 6, 8 } } });
    }

    private int fInput;

    private int fExpected;

    public FibonacciTest(int input, int expected) {
        fInput= input;
        fExpected= expected;
    }

    @Test
    public void test() {
        assertEquals(fExpected, Fibonacci.compute(fInput));
```

```
    }  
}
```

Each instance of `FibonacciTest` will be constructed using the two-argument constructor and the data values in the `@Parameters` method.

static interface	Parameterized.Parameters Annotation for a method which provides parameters to be injected into the test class constructor by Parameterized

Nested classes/interfaces inherited from class org.junit.runners.Suite	
	Suite.SuiteClasses

	Parameterized(Class<?> klass) Only called reflectively.

protected List<Runner>	getChildren() Returns a list of objects that define the children of this Runner.

org.junit.runners. Suite	
	describeChild , emptySuite , runChild

org.junit.runners. ParentRunner	
	childrenInvoker , classBlock , collectInitializationErrors , filter , getDescription , getName , getTestClass , run , setScheduler , sort , validatePublicVoidNoArgMethods , withAfterClasses , withBeforeClasses

org.junit.runner. Runner	

[testCount](#)

java.lang. Object

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#),
[toString](#), [wait](#), [wait](#)

Parameterized

```
public Parameterized(Class<?> klass)
    throws Throwable
```

Only called reflectively. Do not use programmatically.

:

[Throwable](#)

getChildren

```
protected List<Runner> getChildren()
```

Description copied from class: [ParentRunner](#)

Returns a list of objects that define the children of this Runner.

:

[getChildren](#) in class [Suite](#)

... | ...
:| NESTED | .l.

:| .l.



: REQUIRED | OPTIONAL

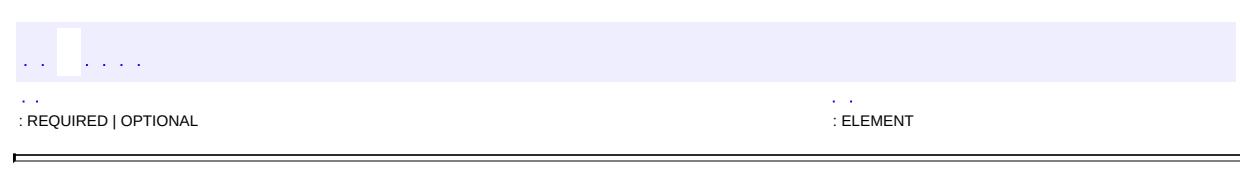
: ELEMENT



org.junit.runners Annotation Type Parameterized.Parameters

```
@Retention(value=RUNTIME)
@Target(value=METHOD)
public static @interface Parameterized.Parameters
```

Annotation for a method which provides parameters to be injected into the test class constructor by Parameterized



The diagram shows the `Parameterized.Parameters` annotation with two small icons below it. On the left is a blue square icon with a white circle and a vertical bar, representing the `REQUIRED` or `OPTIONAL` status. On the right is a blue square icon with a white triangle, representing the `ELEMENT` type.

: REQUIRED | OPTIONAL : ELEMENT



... 1 ...

... 1 ...



org.junit.runners Class ParentRunner<T>

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.runners.ParentRunner<T>
```

:

[Describable](#), [Filterable](#), [Sortable](#)

:

[BlockJUnit4ClassRunner](#), [Suite](#)

```
public abstract class ParentRunner<T>
```

extends [Runner](#)

implements [Filterable](#), [Sortable](#)

Provides most of the functionality specific to a Runner that implements a "parent node" in the test tree, with children defined by objects of some data type T. (For [BlockJUnit4ClassRunner](#), T is [Method](#). For [Suite](#), T is [Class](#).) Subclasses must implement finding the children of the node, describing each child, and running each child. ParentRunner will filter and sort children, handle @BeforeClass and @AfterClass methods, create a composite [Description](#), and run children sequentially.

protected	ParentRunner (Class <?> testClass) Constructs a new ParentRunner that will run @TestClass
-----------	---

protected org.junit.runners.model.Statement	childrenInvoker (RunNotifier notifier) Returns a Statement: Call runChild(Obj each object returned by getChildren() (subject to sort)
--	---

protected org.junit.runners.model.Statement	classBlock(RunNotifier notifier) Constructs a Statement to run all of the tests.
protected void	collectInitializationErrors(List<Throwable Adds to errors a throwable for each problematic class (available from getTestClass()).
protected abstract Description	describeChild(T child) Returns a Description for child, which can be an element of the list returned by getChildren() .
void	filter(Filter filter) Remove tests that don't pass the parameter.
protected abstract List<T>	getChildren() Returns a list of objects that define the children.
Description	getDescription()
protected String	getName() Returns a name used to describe this Runner.
org.junit.runners.model.TestClass	getTestClass() Returns a TestClass object wrapping the class.
void	run(RunNotifier notifier) Run the tests for this runner.
protected abstract void	runChild(T child, RunNotifier notifier) Runs the test corresponding to child, which can be an element of the list returned by getChildren() .
void	setScheduler(org.junit.runners.model.Runn Sets a scheduler that determines the order a children.
void	sort(Sorter sorter) Sorts the tests using sorter.
protected void	validatePublicVoidNoArgMethods(Class<? ex Annotation , boolean isStatic, List<Throwable errors) Adds to errors if any method in this class has the given annotation, but: is not public, or takes parameters other than void, or is static (given isStatic is false) (given isStatic is true).
	withAfterClasses(org.junit.runners.model.

protected
org.junit.runners.model.Statement

Returns a Statement: run all non-overridden methods on this class and superclasses before executing statements. Methods are always executed: exceptions thrown are combined, if necessary, with exceptions from AfterMultipleFailureException.

protected
org.junit.runners.model.Statement

[withBeforeClasses](#)(org.junit.runners.model.Statement) Returns a Statement: run all non-overridden methods on this class and superclasses before executing statements. If an Exception, stop execution and pass the exception.

org.junit.runner. [Runner](#)

[testCount](#)

java.lang. [Object](#)

[clone](#), [equals](#), [finalize](#), [getClass](#), [hashCode](#), [notify](#), [notifyAll](#), [toString](#), [wait](#), [wait](#), [wait](#)

ParentRunner

protected **ParentRunner**([Class](#)<?> testClass)
throws org.junit.runners.model.InitializationError

Constructs a new ParentRunner that will run @TestClass

:

org.junit.runners.model.InitializationError

getChildren

protected abstract [List](#)<T> **getChildren()**

Returns a list of objects that define the children of this Runner.

describeChild

```
protected abstract Description describeChild(I child)
```

Returns a [Description](#) for child, which can be assumed to be an element of the list returned by [getChildren\(\)](#)

runChild

```
protected abstract void runChild(I child,  
                         RunNotifier notifier)
```

Runs the test corresponding to child, which can be assumed to be an element of the list returned by [getChildren\(\)](#). Subclasses are responsible for making sure that relevant test events are reported through notifier

collectInitializationErrors

```
protected void collectInitializationErrors(List<Throwable> errors)
```

Adds to errors a throwable for each problem noted with the test class (available from [getTestClass\(\)](#)). Default implementation adds an error for each method annotated with @BeforeClass or @AfterClass that is not public static void with no arguments.

validatePublicVoidNoArgMethods

```
protected void validatePublicVoidNoArgMethods(Class<? extends Annotation> annotation,  
                          boolean isStatic,  
                          List<Throwable> errors)
```

Adds to errors if any method in this class is annotated with annotation, but:

- is not public, or
- takes parameters, or
- returns something other than void, or

- is static (given `isStatic` is `false`), or
 - is not static (given `isStatic` is `true`).
-

classBlock

```
protected org.junit.runners.model.Statement classBlock(RunNotifier n
```

Constructs a Statement to run all of the tests in the test class. Override to add pre-/post-processing. Here is an outline of the implementation:

- Call [`runChild\(Object, RunNotifier\)`](#) on each object returned by [`getChildren\(\)`](#) (subject to any imposed filter and sort).
- ALWAYS run all non-overridden `@BeforeClass` methods on this class and superclasses before the previous step; if any throws an Exception, stop execution and pass the exception on.
- ALWAYS run all non-overridden `@AfterClass` methods on this class and superclasses before any of the previous steps; all `AfterClass` methods are always executed: exceptions thrown by previous steps are combined, if necessary, with exceptions from `AfterClass` methods into a `MultipleFailureException`.

```
:  
    notifier -  
:  
    Statement
```

withBeforeClasses

```
protected org.junit.runners.model.Statement withBeforeClasses(org.ju
```

Returns a Statement: run all non-overridden `@BeforeClass` methods on this class and superclasses before executing statement; if any throws an Exception, stop execution and pass the exception on.

withAfterClasses

```
protected org.junit.runners.model.Statement withAfterClasses(org.jun
```

Returns a Statement: run all non-overridden @AfterClass methods on this class and superclasses before executing statement; all AfterClass methods are always executed: exceptions thrown by previous steps are combined, if necessary, with exceptions from AfterClass methods into a MultipleFailureException.

childrenInvoker

protected org.junit.runners.model.Statement **childrenInvoker**([RunNotifier](#))

Returns a Statement: Call [runChild\(Object, RunNotifier\)](#) on each object returned by [getChildren\(\)](#) (subject to any imposed filter and sort)

getName

protected [String](#) **getName**()

Returns a name used to describe this Runner

getTestClass

public final org.junit.runners.model.TestClass **getTestClass**()

Returns a TestClass object wrapping the class to be executed.

getDescription

public [Description](#) **getDescription**()

:

[getDescription](#) in interface [Describable](#)

:

[getDescription](#) in class [Runner](#)

:

a [Description](#) showing the tests to be run by the receiver

run

```
public void run(RunNotifier notifier)
```

Description copied from class: [Runner](#)

Run the tests for this runner.

:

[run](#) in class [Runner](#)

:

notifier - will be notified of events while tests are being run--tests
being started, finishing, and failing

filter

```
public void filter(Filter filter)
    throws NoTestsRemainException
```

Description copied from interface: [Filterable](#)

Remove tests that don't pass the parameter filter.

:

[filter](#) in interface [Filterable](#)

:

filter - the [Filter](#) to apply

:

[NoTestsRemainException](#) - if all tests are filtered out

sort

```
public void sort(Sorter sorter)
```

Description copied from interface: [Sortable](#)

Sorts the tests using sorter

:

[sort](#) in interface [Sortable](#)

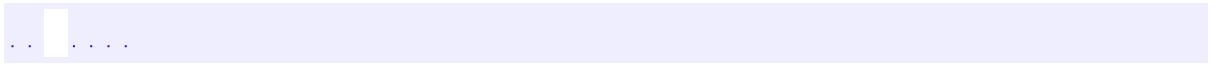
:

sorter - the [Sorter](#) to use for sorting the tests

setScheduler

```
public void setScheduler(org.junit.runners.model.RunnerScheduler sch
```

Sets a scheduler that determines the order and parallelization of children.
Highly experimental feature that may change.



... | ...
:| :|



... [NESTED](#) || .|.

... | .|.



org.junit.runners Class Suite

```
java.lang.Object
└ org.junit.runner.Runner
    └ org.junit.runners.ParentRunner<Runner>
        └ org.junit.runners.Suite
```

:

[Describable](#), [Filterable](#), [Sortable](#)

:

[Parameterized](#)

```
public class Suite

extends ParentRunner<Runner>
```

Using Suite as a runner allows you to manually build a suite containing tests from many classes. It is the JUnit 4 equivalent of the JUnit 3.8.x static `suite()` method. To use it, annotate a class with `@RunWith(Suite.class)` and `@SuiteClasses(TestClass1.class, ...)`. When you run this class, it will run all the tests in all the suite classes.

Suite.SuiteClasses	
static interface	The <code>SuiteClasses</code> annotation specifies the classes to be run when a class annotated with <code>@RunWith(Suite.class)</code> is run.

protected	<code>Suite(Class<?> klass, Class<?>[] suiteClasses)</code> Call this when the default builder is good enough.
protected	<code>Suite(Class<?> klass, List<Runner> runners)</code> Called by this class and subclasses once the runners making up the suite have been determined
	<code>Suite(Class<?> klass,</code>

	<pre>org.junit.runners.model.RunnerBuilder builder)</pre> <p>Called reflectively on classes annotated with <code>@RunWith(Suite.class)</code></p>
	<pre>Suite(org.junit.runners.model.RunnerBuilder builder, Class<?>[] classes)</pre> <p>Call this when there is no single root class (for example, multiple class names passed on the command line to JUnitCore)</p>
protected	<pre>Suite(org.junit.runners.model.RunnerBuilder builder, Class<?> klass, Class<?>[] suiteClasses)</pre> <p>Called by this class and subclasses once the classes making up the suite have been determined</p>

protected Description	<pre>describeChild(Runner child)</pre> <p>Returns a Description for child, which can be assumed to be an element of the list returned by ParentRunner.getChildren()</p>
static Runner	<pre>emptySuite()</pre> <p>Returns an empty suite.</p>
protected List<Runner>	<pre>getChildren()</pre> <p>Returns a list of objects that define the children of this Runner.</p>
protected void	<pre>runChild(Runner runner, RunNotifier notifier)</pre> <p>Runs the test corresponding to child, which can be assumed to be an element of the list returned by ParentRunner.getChildren().</p>

org.junit.runners. ParentRunner	
	childrenInvoker , classBlock , collectInitializationErrors , filter , getDescription , getName , getTestClass , run , setScheduler , sort , validatePublicVoidNoArgMethods , withAfterClasses , withBeforeClasses

org.junit.runner. Runner	
	testCount

--

java.lang. Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,  
toString, wait, wait, wait
```

Suite

```
public Suite(Class<?> klass,  
                 org.junit.runners.model.RunnerBuilder builder)  
throws org.junit.runners.model.InitializationError
```

Called reflectively on classes annotated with @RunWith(Suite.class)

:

klass - the root class

builder - builds runners for classes in the suite

:

org.junit.runners.model.InitializationError

Suite

```
public Suite(org.junit.runners.model.RunnerBuilder builder,  
               Class<?>[] classes)  
throws org.junit.runners.model.InitializationError
```

Call this when there is no single root class (for example, multiple class names passed on the command line to [JUnitCore](#))

:

builder - builds runners for classes in the suite

classes - the classes in the suite

:

org.junit.runners.model.InitializationError

Suite

```
protected Suite(Class<?> klass,
               Class<?>[] suiteClasses)
    throws org.junit.runners.model.InitializationError
```

Call this when the default builder is good enough. Left in for compatibility with JUnit 4.4.

:

 klass - the root of the suite
 suiteClasses - the classes in the suite

:

 org.junit.runners.model.InitializationError

Suite

```
protected Suite(org.junit.runners.model.RunnerBuilder builder,
               Class<?> klass,
               Class<?>[] suiteClasses)
    throws org.junit.runners.model.InitializationError
```

Called by this class and subclasses once the classes making up the suite have been determined

:

 builder - builds runners for classes in the suite
 klass - the root of the suite
 suiteClasses - the classes in the suite

:

 org.junit.runners.model.InitializationError

Suite

```
protected Suite(Class<?> klass,
               List<Runner> runners)
    throws org.junit.runners.model.InitializationError
```

Called by this class and subclasses once the runners making up the suite have been determined

:

klass - root of the suite
runners - for each class in the suite, a [Runner](#)
:
org.junit.runners.model.InitializationError

emptySuite

public static [Runner](#) emptySuite()

Returns an empty suite.

getChildren

protected [List<Runner>](#) getChildren()

Description copied from class: [ParentRunner](#)

Returns a list of objects that define the children of this Runner.

:

[getChildren](#) in class [ParentRunner<Runner>](#)

describeChild

protected [Description](#) describeChild([Runner](#) child)

Description copied from class: [ParentRunner](#)

Returns a [Description](#) for child, which can be assumed to be an element of the list returned by [ParentRunner.getChildren\(\)](#)

:

[describeChild](#) in class [ParentRunner<Runner>](#)

runChild

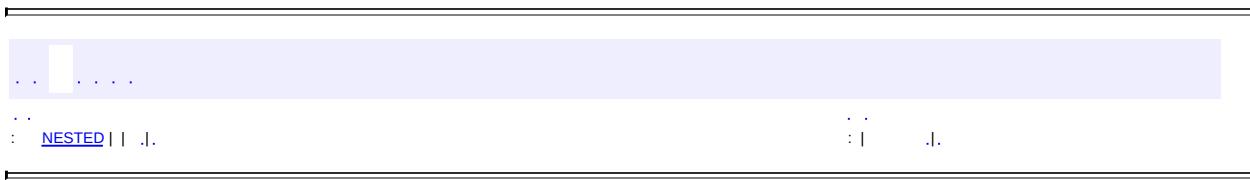
```
protected void runChild(Runner runner,  
RunNotifier notifier)
```

Description copied from class: [ParentRunner](#)

Runs the test corresponding to child, which can be assumed to be an element of the list returned by [ParentRunner.getChildren\(\)](#). Subclasses are responsible for making sure that relevant test events are reported through notifier

:

[runChild](#) in class [ParentRunner<Runner>](#)



CLASS

: [REQUIRED](#) | OPTIONAL

[ELEMENT](#)

org.junit.runners Annotation Type Suite.SuiteClasses

```
@Retention(value=RUNTIME)
@Target(value=TYPE)
@Inherited
public static @interface Suite.SuiteClasses
```

The SuiteClasses annotation specifies the classes to be run when a class annotated with @RunWith(Suite.class) is run.

Required Element Summary

<code>Class<?>[]</code>	<code>value</code>
>[]	

value

public abstract `Class<?>[] value`

:

the classes to be run

...
CLASS : REQUIRED | OPTIONAL ELEMENT

ÊùÓÐÀà [After](#)

[AfterClass](#)
[AllOf](#)
[AllTests](#)
[AnyOf](#)
[Assert](#)
[Assume](#)
[Before](#)
[BeforeClass](#)
[BlockJUnit4ClassRunner](#)
[ComparisonFailure](#)
[Computer](#)
[*Describable*](#)
[DescribedAs](#)
[Description](#)
[Failure](#)
[Filter](#)
[*Iterable*](#)
[Ignore](#)
[Is](#)
[IsAnything](#)
[isEqual](#)
[IsInstanceOf](#)
 [IsNot](#)
[IsNull](#)
[IsSame](#)
[JUnit4](#)
[JUnitCore](#)
[JUnitMatchers](#)
[NoTestsRemainException](#)
[Parameterized](#)
[Parameterized.Parameters](#)
[ParentRunner](#)
[Request](#)
[Result](#)
[Rule](#)
[RunListener](#)
[Runner](#)
[RunNotifier](#)
[RunWith](#)
[*Sortable*](#)
[Sorter](#)
[StoppedByUserException](#)
[Suite](#)
[Suite.SuiteClasses](#)
[Test](#)
[Test.None](#)