After
AfterClass
AllOf
AllTests
AnyOf
Assert
Assume
Before
BeforeClass
BlockJUnit4ClassRunner
ComparisonFailure
Computer
Describable
DescribedAs
Description
Failure
Filter
Filterable
Ignore
Is
IsAnything
IsEqual
IsInstanceOf
IsNot
IsNull
IsSame
JUnit4
JUnitCore
JUnitMatchers
NoTests RemainException
Parameterized
Parameterized. Parameters
ParentRunner
Request
Result
Rule
RunListener
Runner
RunNotifier
RunWith
Sortable
Sorter
StoppedByUserException
Suite
Suite. Suite Classes
Test
Test. None
Deprecated API

- **Deprecated Methods**

<table>
<thead>
<tr>
<th>Deprecated Methods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>org.junit.Assert.assertEquals(double, double)</code></td>
<td><em>Use <code>assertEquals(double expected, double actual, double epsilon)</code> instead</em></td>
</tr>
<tr>
<td><code>org.junit.Assert.assertEquals(Object[], Object[])</code></td>
<td><em>Use <code>assertArrayEquals</code></em></td>
</tr>
<tr>
<td><code>org.junit.Assert.assertEquals(String, double, double)</code></td>
<td><em>Use <code>assertEquals(String message, double expected, double actual, double epsilon)</code> instead</em></td>
</tr>
<tr>
<td><code>org.junit.Assert.assertEquals(String, Object[], Object[])</code></td>
<td><em>Use <code>assertArrayEquals</code></em></td>
</tr>
<tr>
<td><code>org.junit.runner.Request.errorReport(Class, Throwable)</code></td>
<td></td>
</tr>
<tr>
<td><code>org.junit.runners.BlockJUnit4ClassRunner.possiblyExpectingExceptions(FrameworkMethod, Object, Statement)</code></td>
<td><em>Will be private soon: use Rules instead</em></td>
</tr>
<tr>
<td><code>org.junit.runners.BlockJUnit4ClassRunner.validateInstanceMethods(List)</code></td>
<td><em>unused API, will go away in future version</em></td>
</tr>
<tr>
<td><code>org.junit.runners.BlockJUnit4ClassRunner.withAfters(FrameworkMethod, Object, Statement)</code></td>
<td><em>Will be private soon: use Rules instead</em></td>
</tr>
<tr>
<td><code>org.junit.runners.BlockJUnit4ClassRunner.withBefores(FrameworkMethod, Object, Statement)</code></td>
<td><em>Will be private soon: use Rules instead</em></td>
</tr>
<tr>
<td><code>org.junit.runners.BlockJUnit4ClassRunner.withPotentialTimeout(FrameworkMethod, Object, Statement)</code></td>
<td><em>Will be private soon: use Rules instead</em></td>
</tr>
</tbody>
</table>
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

()  

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](https://example.com/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](https://example.com/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
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.notification.RunNotifier
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(Object, Object)** - Static method in class org.junit.Assert
  Asserts that two objects are equal.

**assertEquals(String, 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.
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`

**assertArrayEquals(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.
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.
childlessCopy() - Method in class org.junit.runner.Description

cleanChildrenInvoker(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

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.
**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](https://github.com/junit-team/junit5)

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

**getTrace()** - Method in class org.junit.runner.notification.Failure
   Convenience method

**getVersion()** - Method in class org.junit.runnerJUnitCore
File

**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.

**Is(Matcher<T>)** - Constructor for class `org.hamcrest.core.Is`  
Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive.

**is(Matcher<T>)** - Static method in class `org.hamcrest.core.Is`  
This is a shortcut to the frequently used `is(equalTo(x))`.

**is(T)** - 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`  
A matcher that always returns `true`.

**IsAnything(String)** - Constructor for class `org.hamcrest.core.IsAnything`  
A matcher that always returns `true`.

**isEmpty()** - Method in class `org.junit.runner.Description`  
Tests whether the value is an instance of a class.

**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`  
A matcher that always returns `true`.

**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
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.runnerJUnitCore
   Create a new JUnitCore to run tests.
JUnitMatchers - Class in org.junit.matchers
   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() - Constructor for class org.junit.matchersJUnitMatchers
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.AreEqual
matches(Object) - Method in class org.hamcrest.core.IsNotNull
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
\textbf{not(Matcher<T>)} - Static method in class \texttt{org.hamcrest.core.IsNot}
Inverts the rule.

\textbf{not(T)} - Static method in class \texttt{org.hamcrest.core.IsNot}
This is a shortcut to the frequently used \texttt{not(equalTo(x))}.

\textbf{NoTests RemainException} - Exception in \texttt{org.junit.runner.manipulation}
Thrown when a filter removes all tests from a runner.

\textbf{NoTests RemainException()} - Constructor for exception
\texttt{org.junit.runner.manipulation.NoTests RemainException}

\textbf{notNullValue()} - Static method in class \texttt{org.hamcrest.core.IsNotNull}
Matches if value is not null.

\textbf{notNullValue(Class<T>)} - Static method in class \texttt{org.hamcrest.core.IsNotNull}
Matches if value is not null.

\textbf{NULL} - Static variable in class \texttt{org.junit.runner.manipulation.Sorter}
NULL is a \texttt{Sorter} that leaves elements in an undefined order.

\textbf{nullValue()} - Static method in class \texttt{org.hamcrest.core.IsNotNull}
Matches if value is null.

\textbf{nullValue(Class<T>)} - Static method in class \texttt{org.hamcrest.core.IsNotNull}
Matches if value is null.
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.
**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*
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.runnerJUnitCore`
Run the tests contained in classes.

`runClasses(Class<?>...)` - Static method in class `org.junit.runnerJUnitCore`
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.runnerJUnitCore`
Do not use.

`runMainAndExit(JUnitSystem, String...)` - Static method in class `org.junit.runnerJUnitCore`
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.
**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.
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
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)
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
# JUnit API 4.7

## JUnit 4 API

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

2009 www.junit.org
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.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>
  - 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.runners.**AllTests**
  - org.junit.runners.**ParentRunner**<T> (implements org.junit.runner.manipulation.**Filterable**, org.junit.runner.manipulation.**Sortable**) (implements 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.RuntimeException
    - org.junit.runner.notification.StoppedByUserException
- org.junit.Test.None
Interface Hierarchy

- org.junit.runner.Descrivable
- 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.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
```
**org.hamcrest.core Class AllOf<T>**

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

```

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`.

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

```java
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.

```java
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.

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

Generates a description of the object.

```java
boolean matches(Object o)
```

Evaluates the matcher for argument `item`.

**org.hamcrest.BaseMatcher**

`_dont_implement_Matcher___instead_extend_BaseMatcher_`, `toString`
**java.lang. Object**
close, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

---

**AllOf**

class **AllOf** {Iterable<org.hamcrest.Matcher<? extends T>> matchers)

---

**matches**

class **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**

class **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

```java
public static <T> org.hamcrest.Matcher<T> allOf(org.hamcrest.Matcher<? super T>... matchers)
```

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

---

allOf

```java
public static <T> org.hamcrest.Matcher<T> allOf(Iterable<org.hamcrest.Matcher<? super T>> matchers)
```

Evaluates to true only if ALL of the passed in matchers evaluate to true.
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 T>> 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(Iterable<org.hamcrest.Matcher<? extends T>>... matchers)
  Evaluates to true if ANY of the passed in matchers evaluate to true.

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

boolean matches(Object o)
    Evaluates the matcher for argument item.
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)

```java
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<? super T>... matchers)

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

---

anyOf


Evaluates to true if ANY of the passed in matchers evaluate to true.
org.hamcrest.core Class DescribedAs<T>

java.lang.Object
downcast org.hamcrest.BaseMatcher<T>
downcast 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)

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

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

describeTo(org.hamcrest.Description description)

Generates a description of the object.

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<? super 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 description, org.hamcrest.Matcher<T> matcher, Object... values)
```

Wraps an existing matcher and overrides the description when it fails.
**org.hamcrest.core** Class *Is<T>*

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


```java
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)))`
```

---

```java
| is(org.hamcrest.Matcher<T> matcher) |
```

---

```java
| void describeTo(org.hamcrest.Description descr) |
| Generates a description of the object. |
```

```java
| static org.hamcrest.Matcher<Object> is(Class<?> type) |
| This is a shortcut to the frequently used `is(instanceOf(SomeClass.class))`. |
```

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

```java
| is(org.hamcrest.Matcher<T> matcher) |
| Decorates another Matcher, retaining the behavior but allowing tests to be slightly more expressive. |
```

```java
| static <T> org.hamcrest.Matcher<T> |
```
**is** (<code>T value</code>)

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

**matches** (<code>Object arg</code>)

Evaluates the matcher for argument <code>item</code>.

```java
public boolean matches(Object arg)
```

**Description copied from interface: org.hamcrest.Matcher**

Evaluates the matcher for argument <code>item</code>.

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.

- <code>arg</code> - the object against which the matcher is evaluated.
- <code>true</code> if <code>item</code> matches, otherwise <code>false</code>. 
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> matcher)

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))
org.hamcrest.core  **Class IsAnything<T>**

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

:  
```

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

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

A matcher that always returns true.
```

```java
IsAnything()
```

```java
IsAnything(String description)
```

```java
static <T> org.hamcrest.Matcher<T> any(Class<T> type)
    This matcher always evaluates to true.
```

```java
static <T> org.hamcrest.Matcher<T> anything()
    This matcher always evaluates to true.
```

```java
static <T> org.hamcrest.Matcher<T> anything(String description)
    This matcher always evaluates to true.
```

```java
void describeTo(org.hamcrest.Description description)```
Generates a description of the object.  boolean `matches(Object o)`

Evaluates the matcher for argument *item*.

```java
org.hamcrest.BaseMatcher
_dont_implement_Matcher___instead_extend_BaseMatcher_, toString
```

```java
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.
org.hamcrest.core  Class IsEqual<T>

java.lang.Object
   ▼ org.hamcrest.BaseMatcher<T>
      ▼ org.hamcrest.core.IsEqual<T>

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

public class IsEqual<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)

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

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

matches(Object arg)
Evaluates the matcher for argument item.

dontImplement_Matcher__instead_extend_BaseMatcher__, toString
java.lang.  **Object**
close, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

**IsEqual**

class **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**

```java
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?
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

---

| 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 item. |

---

| org.hamcrest.BaseMatcher |
| _dont_implement_Matcher___instead_extend_BaseMatcher_, toString |

---

| java.lang. Object |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait |
**IsInstanceOf**

public IsInstanceOf(Class<? extends 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
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.

---

**instanceOf**

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

Is the value an instance of a particular type?
Class IsNot<T>

extends org.hamcrest.BaseMatcher<T>

Calculates the logical negation of a matcher.

public class IsNot<T>

not(org.hamcrest.Matcher<T> matcher)

Inverts the rule.

not(T value)

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

public IsNot(org.hamcrest.Matcher<? super T> matcher)

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**

```java
class org.hamcrest.Matcher<T> { public static <T> Matcher<T> not(Matcher<T> matcher) { ... } }
```
Inverts the rule.

---

**not**

```java
class org.hamcrest.Matcher<T> { public static <T> 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)))`
**org.hamcrest.core Class IsNull<T>**

```java
java.lang.Object
    ↓ org.hamcrest.BaseMatcher<T>
       ↓ org.hamcrest.core.IsNull<T>

```

```java
public class IsNull<T>

extends org.hamcrest.BaseMatcher<T>

Is the value null?
```

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsNull()</code></td>
<td></td>
</tr>
<tr>
<td><code>describeTo(Description description)</code></td>
<td>Generates a description of the object.</td>
</tr>
<tr>
<td><code>matches(Object o)</code></td>
<td>Evaluates the matcher for argument <em>item</em>.</td>
</tr>
<tr>
<td><code>notNullValue()</code></td>
<td>Matches if value is not null.</td>
</tr>
</tbody>
</table>

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

Matches if value is not null.
```

```java
static <T> org.hamcrest.Matcher<T> notNullValue(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.
```
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.
org.hamcrest.core Class IsSame<T>

java.lang.Object
   ▼ org.hamcrest.BaseMatcher<T>
      ▼ org.hamcrest.core.IsSame<T>


public class IsSame<T>
extends org.hamcrest.BaseMatcher<T>

Is the value the same object as another value?

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsSame(T object)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>void describeTo(org.hamcrest.Description description)</td>
</tr>
<tr>
<td>Generates a description of the object.</td>
</tr>
<tr>
<td>boolean matches(Object arg)</td>
</tr>
<tr>
<td>Evaluates the matcher for argument item.</td>
</tr>
<tr>
<td>static sameInstance(T object)</td>
</tr>
<tr>
<td>Creates a new instance of IsSame</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.hamcrest.BaseMatcher</td>
</tr>
<tr>
<td><em>dont_implement_Matcher__instead_extend_BaseMatcher</em>, toString</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang. Object</td>
</tr>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll,</td>
</tr>
</tbody>
</table>
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)
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.

---
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 `Object.equals(java.lang.Object)` 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.
Hierarchy For Package org.hamcrest.core

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.hamcrest.BaseMatcher&lt;T&gt; (implements org.hamcrest.Matcher&lt;T&gt;)
    - org.hamcrest.core.**AllOf**&lt;T&gt;
    - org.hamcrest.core.**AnyOf**&lt;T&gt;
    - org.hamcrest.core.**DescribedAs**&lt;T&gt;
    - org.hamcrest.core.**Is**&lt;T&gt;
    - org.hamcrest.core.**IsAnything**&lt;T&gt;
    - org.hamcrest.core.**IsEqual**&lt;T&gt;
    - org.hamcrest.core.**IsInstanceOf**
    - org.hamcrest.core.**IsNot**&lt;T&gt;
    - org.hamcrest.core.**IsNull**&lt;T&gt;
    - org.hamcrest.core.**IsSame**&lt;T&gt;
@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:

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

: BEFORE, TEST
<table>
<thead>
<tr>
<th>REQUIRED</th>
<th>OPTIONAL</th>
<th>ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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:

```java
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
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

<p>| 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) |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static void <code>assertArrayEquals(int[] expecteds, int[] actuals)</code></td>
<td>Asserts that two int arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(long[] expecteds, long[] actuals)</code></td>
<td>Asserts that two long arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(Object[] expecteds, Object[] actuals)</code></td>
<td>Asserts that two object arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(short[] expecteds, short[] actuals)</code></td>
<td>Asserts that two short arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, byte[] expecteds, byte[] actuals)</code></td>
<td>Asserts that two byte arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, char[] expecteds, char[] actuals)</code></td>
<td>Asserts that two char arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, double[] expecteds, double[] actuals, double delta)</code></td>
<td>Asserts that two double arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, float[] expecteds, float[] actuals, float delta)</code></td>
<td>Asserts that two float arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, int[] expecteds, int[] actuals)</code></td>
<td>Asserts that two int arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, long[] expecteds, long[] actuals)</code></td>
<td>Asserts that two long arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, Object[] expecteds, Object[] actuals)</code></td>
<td>Asserts that two object arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertArrayEquals(String message, short[] expecteds, short[] actuals)</code></td>
<td>Asserts that two short arrays are equal.</td>
</tr>
<tr>
<td>static void <code>assertEquals(double expected, double actual)</code></td>
<td><strong>Deprecated. Use assertEquals(double expected, double actual, double epsilon) instead</strong></td>
</tr>
<tr>
<td>static void <code>assertEquals(double expected, double actual, double epsilon)</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>static void double delta)</td>
<td>Asserts that two doubles or floats are equal to within a positive delta.</td>
</tr>
<tr>
<td>static void assertEquals(long expected, long actual)</td>
<td>Asserts that two longs are equal.</td>
</tr>
<tr>
<td>static void assertEquals(Object[] expecteds, Object[] actuals)</td>
<td>Deprecated. Use assertArrayEquals instead.</td>
</tr>
<tr>
<td>static void assertEquals(Object expected, Object actual)</td>
<td>Asserts that two objects are equal.</td>
</tr>
<tr>
<td>static void assertEquals(String message, double expected, double actual)</td>
<td>Deprecated. Use assertEquals(String message, double expected, double actual, double epsilon) instead.</td>
</tr>
<tr>
<td>static void assertEquals(String message, double expected, double actual, double delta)</td>
<td>Asserts that two doubles or floats are equal to within a positive delta.</td>
</tr>
<tr>
<td>static void assertEquals(String message, long expected, long actual)</td>
<td>Asserts that two longs are equal.</td>
</tr>
<tr>
<td>static void assertEquals(String message, Object[] expecteds, Object[] actuals)</td>
<td>Deprecated. use assertArrayEquals</td>
</tr>
<tr>
<td>static void assertEquals(String message, Object expected, Object actual)</td>
<td>Asserts that two objects are equal.</td>
</tr>
<tr>
<td>static void assertFalse(boolean condition)</td>
<td>Asserts that a condition is false.</td>
</tr>
<tr>
<td>static void assertFalse(String message, boolean condition)</td>
<td>Asserts that a condition is false.</td>
</tr>
<tr>
<td>static void assertNotNull(Object object)</td>
<td>Asserts that an object isn't null.</td>
</tr>
<tr>
<td>static void assertNotNull(String message, Object object)</td>
<td>Asserts that an object isn't null.</td>
</tr>
<tr>
<td>static void assertNotNullSame(Object unexpected, Object actual)</td>
<td>Asserts that two objects do not refer to the same object.</td>
</tr>
<tr>
<td>static void assertNotNullSame(String message, Object unexpected,</td>
<td>Asserts that two objects do not refer to the same object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>static void Object actual)</td>
<td>Asserts that two objects do not refer to the same object.</td>
</tr>
<tr>
<td>static void assertNotNull(Object object)</td>
<td>Asserts that an object is null.</td>
</tr>
<tr>
<td>static void assertNotNull(String message, Object object)</td>
<td>Asserts that an object is null.</td>
</tr>
<tr>
<td>static void assertNull(Object object)</td>
<td>Asserts that an object is null.</td>
</tr>
<tr>
<td>static void assertNull(String message, Object object)</td>
<td>Asserts that an object is null.</td>
</tr>
<tr>
<td>static void assertSame(Object expected, Object actual)</td>
<td>Asserts that two objects refer to the same object.</td>
</tr>
<tr>
<td>static void assertSame(String message, Object expected, Object actual)</td>
<td>Asserts that two objects refer to the same object.</td>
</tr>
<tr>
<td>static &lt;T&gt; void assertThat(String reason, T actual, org.hamcrest.Matcher&lt;T&gt; matcher)</td>
<td>Asserts that actual satisfies the condition specified by matcher.</td>
</tr>
<tr>
<td>static &lt;T&gt; void assertThat(T actual, org.hamcrest.Matcher&lt;T&gt; matcher)</td>
<td>Asserts that actual satisfies the condition specified by matcher.</td>
</tr>
<tr>
<td>static void assertTrue(boolean condition)</td>
<td>Asserts that a condition is true.</td>
</tr>
<tr>
<td>static void assertTrue(String message, boolean condition)</td>
<td>Asserts that a condition is true.</td>
</tr>
<tr>
<td>static void fail()</td>
<td>Fails a test with no message.</td>
</tr>
<tr>
<td>static void fail(String message)</td>
<td>Fails a test with the given message.</td>
</tr>
</tbody>
</table>

### java.lang. Object
- clone
- equals
- finalize
- getClass
- hashCode
- notify
- notifyAll
- toString
- wait
- wait
- wait

**Assert**

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

```java
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
```

```java
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
```

```java
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)
```
okay)
condition - condition to be checked

---

**assertFalse**

```java
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**

```java
public static void fail(String message)
```

Fails a test with the given message.

: message - the identifying message for the `AssertionError` (null okay)

: `AssertionError`

---

**fail**

```java
public static void fail()
```

Fails a test with no message.

---

**assertEquals**

```java
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.

```java
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.

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

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

---

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

Throws `org.junit.internal.ArrayComparisonFailure` if the two object arrays are not equal. If they are not, an `AssertionError` is thrown with the given message. If expecteds and actuals are null, they are considered equal.

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

: 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.ArrayComparationFailure

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
ok)
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 IllegalArgumentException 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.ArrayComparisonFailure

Asserts that two char arrays are equal. If they are not, an IllegalArgumentException is thrown with the given message.

: message - the identifying message for the IllegalArgumentException (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 IllegalArgumentException is thrown.

:
assertArrayEquals

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

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

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

```java
assertArrayEquals()
```

`assertArrayEquals(int[] expecteds, int[] actuals)`

Asserts that two int arrays are equal. If they are not, an `AssertionError` is thrown.

```java
expecteds - int array with expected values.
actuals - int array with actual values
```

```
assertArrayEquals
```

`assertArrayEquals(String message, long[] expecteds, long[] actuals)`

Asserts that two long arrays are equal. If they are not, an `AssertionError` is thrown with the given message.

```java
message - the identifying message for the AssertionError (null okay)
expecteds - long array with expected values.
```
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.ArrayComparisonFailure

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.ArrayComparisonFailure

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.
actualls - float array with actual values

---

**assertEquals**

```java
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**

```java
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

@Deprecated
public static void assertEquals(double expected, double actual)

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

@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.

```
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.

```
message - the identifying message for the `AssertionError` (null okay)
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.

```
object - Object to check or null
```

---

**assertNull**

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

Asserts that an object isn't null. If it is an `AssertionError` is thrown.

```
message - the identifying message for the `AssertionError` (null okay)
object - Object to check or null
```
Asserts that an object is null. If it is not, an **AssertionError** is thrown with the given message.

: 
  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.

: 
  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.

: 
  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**

```java
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**

```java
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.

: `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`

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

**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.

: `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**

public static `<T> void` `assertThat(T actual, org.hamcrest.Matcher<T> matcher)`
Asserts that actual satisfies the condition specified by matcher. If not, an
**AssertionError** is thrown with information about the matcher and failing
value. Example:

```java
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

---

**assertThat**

```java
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:

```java
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
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:

```java
// only provides information if database is reachable.
@Test public void calculateTotalSalary() {
    DBConnection dbc = Database.connect();
    assumeNotNull(dbc);
    // ...
}
```

These methods can be used directly: `Assume.assumeTrue(...)`, however, they read better if they are referenced through static import:

```java
import static org.junit.Assume.*;
...
assumeTrue(...);
```

### Assume ()

Use to assume that an operation completes normally.

```java
static void assumeNoException(Throwables t)
```
### assumeNotNull

```java
public static void assumeNotNull(Object... objects)
```

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

### assumeThat

```java
public static void assumeThat(T actual, org.hamcrest.Matcher<T> matcher)
```

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

### assumeTrue

```java
public static void assumeTrue(boolean b)
```

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

### Assume

#### Public

```java
public Assume()
```

#### Public static void

```java
public static void assumeTrue(boolean b)
```

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

```java
: b -
```

### assumeNotNull

```java
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**

```java
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:

:  
  ```java
  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](https://hamcrest.org/1.3/junit-matchers)

---

**assumeNoException**

```java
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:

```java
@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
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:

```java
public class Example {
    List empty;
    @Before public void initialize() {
        empty = new ArrayList();
    }
    @Test public void size() {
        ...
    }
    @Test public void remove() {
        ...
    }
}
```

: **BeforeClass, After**
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 the current class.

For example:

```java
public class Example {
    @BeforeClass public static void onlyOnce() {
        ...
    }
    @Test public void one() {
        ...
    }
    @Test public void two() {
        ...
    }
}
```

AfterClass
org.junit  Class ComparisonFailure

java.lang.Object  
  ↓ java.lang.Throwables
    ↓ 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.

---

<table>
<thead>
<tr>
<th>String</th>
<th>getActual()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the actual string value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getExpected()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns the expected string value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>String</th>
<th>getMessage()</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns &quot;...&quot; in place of common prefix and &quot;...&quot; in place of common suffix between expected and actual.</td>
</tr>
</tbody>
</table>
### `java.lang. Throwable`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fillInStackTrace</code></td>
<td>Replaces the stack trace with the stack trace of the current exception.</td>
</tr>
<tr>
<td><code>getCause</code></td>
<td>Returns the cause of this exception.</td>
</tr>
<tr>
<td><code>getLocalizedMessage</code></td>
<td>Returns the localised message of this exception.</td>
</tr>
<tr>
<td><code>getStackTrace</code></td>
<td>Returns the stack trace of this exception.</td>
</tr>
<tr>
<td><code>initCause</code></td>
<td>Sets the cause of this exception and re-throws it.</td>
</tr>
<tr>
<td><code>printStackTrace</code></td>
<td>Prints the stack trace of this exception to the standard error stream.</td>
</tr>
<tr>
<td><code>printStackTrace</code></td>
<td>Prints the stack trace of this exception to the specified error stream.</td>
</tr>
<tr>
<td><code>printStackTrace</code></td>
<td>Prints the stack trace of this exception to the specified print stream.</td>
</tr>
<tr>
<td><code>setStackTrace</code></td>
<td>Sets the stack trace of this exception.</td>
</tr>
<tr>
<td><code>toString</code></td>
<td>Returns a string representation of this exception.</td>
</tr>
</tbody>
</table>

### `java.lang. Object`

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>clone</code></td>
<td>Creates a shallow copy of this object.</td>
</tr>
<tr>
<td><code>equals</code></td>
<td>Compares two objects for equality.</td>
</tr>
<tr>
<td><code>finalize</code></td>
<td>Performs any necessary cleanup before an object is reclaimed.</td>
</tr>
<tr>
<td><code>getClass</code></td>
<td>Returns the class of the specified object.</td>
</tr>
<tr>
<td><code>hashCode</code></td>
<td>Returns a hash code for this object.</td>
</tr>
<tr>
<td><code>notify</code></td>
<td>Causes another object in synchronization context to become operational.</td>
</tr>
<tr>
<td><code>notifyAll</code></td>
<td>Causes all objects in a synchronization context to become operational.</td>
</tr>
<tr>
<td><code>wait</code></td>
<td>Causes the current thread of execution to enter an infinite wait state.</td>
</tr>
<tr>
<td><code>wait</code></td>
<td>Causes the current thread of execution to enter a wait state.</td>
</tr>
<tr>
<td><code>wait</code></td>
<td>Causes the current thread of execution to enter a wait state with an optional timeout.</td>
</tr>
</tbody>
</table>

### `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.
Throws\ldots

<table>
<thead>
<tr>
<th>getActual</th>
</tr>
</thead>
</table>

getActual

public \texttt{String getActual()} 

Returns the actual string value

: 
the actual string value

---

getExpected

public \texttt{String getExpected()} 

Returns the expected string value

: 
the expected string value

---
org.junit **Annotation** Type **Ignore**

```java
@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:

```java
@Ignore @Test public void something() { ... }
```

`@Ignore` takes an optional default parameter if you want to record why a test is being ignored:

```java
@Ignore("not ready yet") @Test public void something() { ... }
```

`@Ignore` can also be applied to the test class:

```java
@Ignore public class IgnoreMe {
    @Test public void test1() { ... }
    @Test public void test2() { ... }
}
```

---

**Optional Element Summary**

<table>
<thead>
<tr>
<th><strong>String value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>The optional reason why the test is ignored.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Assert</th>
<th>A set of assertion methods useful for writing tests.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assume</td>
<td>A set of methods useful for stating assumptions about the conditions in which a test is meaningful.</td>
</tr>
<tr>
<td>Test.None</td>
<td>Default empty exception</td>
</tr>
</tbody>
</table>

**Error Summary**

| ComparisonFailure | Thrown when an assertEquals(String, String) fails. |

**Annotation Types Summary**

<table>
<thead>
<tr>
<th>After</th>
<th>If you allocate external resources in a Before method you need to release them after the test runs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterClass</td>
<td>If you allocate expensive external resources in a BeforeClass method you need to release them after all the tests in the class have run.</td>
</tr>
<tr>
<td>Before</td>
<td>When writing tests, it is common to find that several tests need similar objects created before they can run.</td>
</tr>
<tr>
<td>BeforeClass</td>
<td>Sometimes several tests need to share computationally expensive setup (like logging into a database).</td>
</tr>
<tr>
<td>Ignore</td>
<td>Sometimes you want to temporarily disable a test or a group of tests.</td>
</tr>
<tr>
<td>Rule</td>
<td>Annotates fields that contain rules.</td>
</tr>
<tr>
<td>Test</td>
<td>The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case.</td>
</tr>
</tbody>
</table>
Package org.junit Description

Provides JUnit core classes and annotations. Corresponds to junit.framework in Junit 3.x.

: 4.0
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)
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 |
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:

```java
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:

```java
@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:

```java
@Test(timeout=100) public void infinity() {
    while(true);
}
```
### Optional Element Summary

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>expected</strong></td>
<td>Optionally specify expected, a Throwable, to cause a test method to succeed iff an exception of the specified class is thrown by the method.</td>
</tr>
<tr>
<td><strong>timeout</strong></td>
<td>Optionally specify timeout in milliseconds to cause a test method to fail if it takes longer than that number of milliseconds.</td>
</tr>
</tbody>
</table>

#### expected

```java
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.

```java
: org.junit.Test.None.class
```

#### timeout

```java
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.

```java
: 0L
```
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

:\n
Serializable Form

| java.lang.  Throwable |
| fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, setStackTrace, toString |

| java.lang.  Object |
| clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait |

..
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()

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)
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)

<table>
<thead>
<tr>
<th>java.lang. Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait</td>
</tr>
</tbody>
</table>

JUnitMatchers

public JUnitMatchers()

hasItem

public static <T> org.hamcrest.Matcher<Iterable<T>> hasItem(T element)

: element -

A matcher matching any collection containing element
public static <T> org.hamcrest.Matcher<Iterable<T>> hasItem(org.hamcrest.Matcher<? super T> elementMatcher)
  
  : elementMatcher -
  :
  
  A matcher matching any collection containing an element matching elementMatcher

hasItems

public static <T> org.hamcrest.Matcher<Iterable<T>> hasItems(T... elements)

  : elements -
  :
  
  A matcher matching any collection containing every element in elements

hasItems

public static <T> org.hamcrest.Matcher<Iterable<T>> hasItems(org.hamcrest.Matcher<? super T>... elementMatchers)

  : 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.hamcrest.Matcher<T> elementMatcher)

  : elementMatcher -
  :
  
  A matcher matching any collection in which every element matches
containsString

public static org.hamcrest.Matcher<String> containsString(String substring):
    substring -
    a matcher matching any string that contains substring

both

public static <T> org.junit.internal.matchers.CombinableMatcher<T> both:
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> either:
This is useful for fluently combining matchers where either may pass, for example:
    assertThat(string, both(containsString("a")).and(containsStri
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
Hierarchy For Package org.junit.matchers

Package Hierarchies:

All Packages
Class Hierarchy

- java.lang.**Object**
  - org.junit.matchers.**JUnitMatchers**
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.

protected Runner getRunner(org.junit.runners.model.RunnerBuilder builder, Class<?> testClass)
Create a single-class runner for testClass, using builder

protected 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
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 builder, 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.runnersJUnit38ClassRunner, JUnit4, Parameterized,
   ParentRunner, Runner, Suite, org.junit.internal.runners.SuiteMethod

public interface Describable

Represents an object that can describe itself

<table>
<thead>
<tr>
<th>Description</th>
<th>getDescription()</th>
</tr>
</thead>
</table>

getDescription

getDescription():
   a Description showing the tests to be run by the receiver
public class Description 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

<table>
<thead>
<tr>
<th>static Description</th>
<th>EMPTY</th>
<th>Describes a Runner which runs no tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Description</td>
<td>TEST_MECHANISM</td>
<td>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)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td><code>addChild(Description description)</code></td>
<td>Add Description as a child of the receiver.</td>
<td></td>
</tr>
<tr>
<td><code>childlessCopy()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>createSuiteDescription(Class&lt;?&gt; testClass)</code></td>
<td>Create a Description named after testClass</td>
<td></td>
</tr>
<tr>
<td><code>createSuiteDescription(String name, Annotation... annotations)</code></td>
<td>Create a Description named name.</td>
<td></td>
</tr>
<tr>
<td><code>createTestDescription(Class&lt;?&gt; clazz, String name)</code></td>
<td>Create a Description of a single test named name in the class clazz.</td>
<td></td>
</tr>
<tr>
<td><code>createTestDescription(Class&lt;?&gt; clazz, String name, Annotation... annotations)</code></td>
<td>Create a Description of a single test named name in the class clazz.</td>
<td></td>
</tr>
<tr>
<td><code>equals(Object obj)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getAnnotation(Class&lt;T&gt; annotationType)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getAnnotations()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getChildren()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getClass()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getDisplayname()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getMethodname()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getTestclass()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>hashCode()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>isEmpty()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>isSuite()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>isTest()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>testCount()</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>toString()</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`java.lang. Object`
**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**
Create a Description of a single test named name in the class clazz. Generally, this will be a leaf Description.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clazz</td>
<td>the class of the test</td>
</tr>
<tr>
<td>name</td>
<td>the name of the test (a method name for test annotated with Test)</td>
</tr>
<tr>
<td>annotations</td>
<td>meta-data about the test, for downstream interpreters</td>
</tr>
</tbody>
</table>

---

**createTestDescription**

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)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clazz</td>
<td>the class of the test</td>
</tr>
<tr>
<td>name</td>
<td>the name of the test (a method name for test annotated with Test)</td>
</tr>
</tbody>
</table>

---

**createSuiteDescription**

Create a Description named after testCase

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
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**

```java
class isEmpty()

: true if this is a description of a Runner that runs no tests
```

**childlessCopy**

```java
class childlessCopy()

: a copy of this description, with no children (on the assumption that some of the children will be added back)
```

**getAnnotation**

```java
class getAnnotation(Class<T> annotationType)

: the annotation of type annotationType that is attached to this description node, or null if none exists
```

**getAnnotations**

```java
class getAnnotations()

: all of the annotations attached to this description node
```

**getTestClass**

```java
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)
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

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUnitCore()</td>
<td>Create a new JUnitCore to run tests.</td>
</tr>
<tr>
<td>void addListener(RunListener listener)</td>
<td>Add a listener to be notified as the tests run.</td>
</tr>
<tr>
<td>String getVersion()</td>
<td></td>
</tr>
<tr>
<td>static void main(String... args)</td>
<td>Run the tests contained in the classes named in the args.</td>
</tr>
<tr>
<td>void removeListener(RunListener listener)</td>
<td>Remove a listener.</td>
</tr>
<tr>
<td>Result run(Class&lt;?... classes)</td>
<td>Run all the tests in classes.</td>
</tr>
</tbody>
</table>
Result

run(Computer computer, Class<?>... classes)
Run all the tests in classes.

Result

run(Request request)
Run all the tests contained in request.

Result

run(Runner runner)
Do not use.

Result

run(junit.framework.Test test)
Run all the tests contained in JUnit 3.8.x test.

static Result

runClasses(Class<?>... classes)
Run the tests contained in classes.

static Result

runClasses(Computer computer, Class<?>... classes)
Run the tests contained in classes.

Result

runMain(org.junit.internal.JUnitSystem system, String... args)
Do not use.

static void

runMainAndExit(org.junit.internal.JUnitSystem system, String... args)
Do not use.

java.lang. Object

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

gerVersion

public String gerVersion()

: 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**

```java
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**

```java
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**

```java
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

½Ó¿Ú 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.

<table>
<thead>
<tr>
<th>Describable</th>
<th>Represents an object that can describe itself</th>
</tr>
</thead>
</table>

| 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
Hierarchy For Package org.junit.runner

Package Hierarchies:
   All Packages
Class Hierarchy

- java.lang.**Object**
  - org.junit.runner.**Computer**
  - 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)
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 Tests. 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.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static <code>Request</code></td>
<td>Create a Request that, when processed, will run all the tests in a class.</td>
</tr>
<tr>
<td>static <code>Request</code> errorReport(Class&lt;?&gt; klass, Throwable cause)</td>
<td>Deprecated.</td>
</tr>
<tr>
<td>static <code>Request</code> filterWith(Description desiredDescription)</td>
<td>Returns a Request that only runs contains tests whose Description equals desiredDescription</td>
</tr>
<tr>
<td>static <code>Request</code> filterWith(Filter filter)</td>
<td>Returns a Request that only contains those tests that should run when filter is applied</td>
</tr>
<tr>
<td>abstract <code>Runner</code> getRunner()</td>
<td>Returns a <code>Runner</code> for this Request</td>
</tr>
<tr>
<td>static <code>Request</code> method(Class&lt;?&gt; clazz, String methodName)</td>
<td>Create a Request that, when processed, will run a single test.</td>
</tr>
<tr>
<td>static <code>Request</code> runner(Runner runner)</td>
<td></td>
</tr>
<tr>
<td>static <code>Request</code> sortWith(Comparator&lt;Description&gt; comparator)</td>
<td>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:</td>
</tr>
</tbody>
</table>

```java
java.lang. Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

**Request**

```java
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:

```java
private static Comparator forward() {
    return new Comparator() {
        public int compare(Description o1, Description o2) {
            return o1.getDisplayName().compareTo(o2.getDisplayName());
        }
    };
}

public static main() {
    new JUnitCore().run(Request.aClass(AllTests.class).sortWith(forward()));
}
```

comparator - definition of the order of the tests in this Request

a Request with ordered Tests
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.
### java.lang. **Object**

| clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait |

---

### Result

```java
public Result()
```

---

#### getRunCount

```java
public int getRunCount()
```

: the number of tests run

---

#### getFailureCount

```java
public int getFailureCount()
```

: the number of tests that failed during the run

---

#### getRunTime

```java
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.
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

**getDefinition**

```java
public abstract Description getDefinition()
```

:  
  `getDefinition` in interface `Describable`

:  
  a `Description` showing the tests to be run by the receiver

**run**

```java
public abstract void run(RunNotifier notifier)
```

:  
  Run the tests for this runner.

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getDescription</code></td>
<td></td>
</tr>
<tr>
<td><code>run(RunNotifier notifier)</code></td>
<td>Run the tests</td>
</tr>
<tr>
<td><code>testCount</code></td>
<td></td>
</tr>
</tbody>
</table>

java.lang. **Object**

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait`
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
<table>
<thead>
<tr>
<th>CLASS</th>
<th>REQUIRED</th>
<th>OPTIONAL</th>
<th>ELEMENT</th>
</tr>
</thead>
</table>

...
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

<table>
<thead>
<tr>
<th>Class&lt;? extends Runner&gt;</th>
<th>value</th>
</tr>
</thead>
</table>

value

public abstract Class<? extends Runner> value

: a Runner class (must have a constructor that takes a single Class to run)
<table>
<thead>
<tr>
<th>CLASS</th>
<th>REQUIRED</th>
<th>OPTIONAL</th>
<th>ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>static Filter ALL</td>
<td>A null Filter that passes all tests through.</td>
</tr>
</tbody>
</table>

Filter()

apply(Object child)

Invoke with a Runner to cause all tests it intends to run to first be checked with the filter.

describe()

Returns a textual description of this Filter

matchMethodDescription(Description desiredDescription)

Returns a Filter that only runs the single method described by desiredDescription
| abstract boolean | `shouldRun(Description description)` |

```java
java.lang. Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

### ALL

```java
public static Filter ALL
```

A null Filter that passes all tests through.

### Filter

```java
public Filter()
```

### matchMethodDescription

```java
public static Filter matchMethodDescription(Description desiredDescription)
```

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

### shouldRun

```java
public abstract boolean shouldRun(Description description)
```

:
description - the description of the test to be run :
true if the test should be run

---

**describe**

describe

```java
public abstract String describe()
```

Returns a textual description of this Filter :

- a textual description of this Filter

---

**apply**

```java
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
public interface `Filterable`

Runners that allow filtering should implement this interface. Implement `filter(Filter)` to remove tests that don't pass the filter.

```java
void filter(Filter filter)
```

Remove tests that don't pass the parameter filter.

```java
void filter(Filter filter)
    throws NoTestsRemainException
```

Remove tests that don't pass the parameter filter.

```java
    : filter - the Filter to apply
    : NoTestsRemainException - if all tests are filtered out
```
Class NoTestsRemainException

Public class NoTestsRemainException

Extends Exception

Thrown when a filter removes all tests from a runner.

Serializes Form

NoTestsRemainException()
NoTestsRemainException

public NoTestsRemainException()
org.junit.runner.manipulation

Filterable
Sortable

Filter
Sorter

NoTestsRemainException
Package org.junit.runner.manipulation

Provides classes to \texttt{filter} or \texttt{sort} tests.

:  

<table>
<thead>
<tr>
<th>Filterable</th>
<th>Runners that allow filtering should implement this interface.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sortable</td>
<td>Interface for runners that allow sorting of tests.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filter</th>
<th>The canonical case of filtering is when you want to run a single test method in a class.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorter</td>
<td>A Sorter orders tests.</td>
</tr>
</tbody>
</table>

**Exception Summary**

| NoTests RemainException | 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
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
org.junit.runner.manipulation **Interface** Sortable

**All Known Implementing Classes:**
- AllTests, BlockJUnit4ClassRunner,
  org.junit.internal.runnersJUnit38ClassRunner, JUnit4, Parameterized,
  ParentRunner, Suite, org.junit.internal.runners.SuiteMethod

```java
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.

```java
void sort(Sorter sorter)
```

Sorts the tests using sorter

```java
void sort(Sorter sorter)
```

Sorts the tests using sorter

: 
  sorter - the Sorter to use for sorting the tests
Class Sorter

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)
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>
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.

<table>
<thead>
<tr>
<th>Description</th>
<th>getException()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Throwable</td>
<td>getDescription()</td>
</tr>
<tr>
<td>String</td>
<td>getMessage()</td>
</tr>
<tr>
<td>String</td>
<td>getTestHeader()</td>
</tr>
<tr>
<td>String</td>
<td>getTrace()</td>
</tr>
<tr>
<td>String</td>
<td>toString()</td>
</tr>
</tbody>
</table>
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
Package org.junit.runner.notification

Provides information about a test run.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failure</strong></td>
<td>A Failure holds a description of the failed test and the exception that was thrown while running it.</td>
</tr>
<tr>
<td><strong>RunListener</strong></td>
<td>If you need to respond to the events during a test run, extend RunListener and override the appropriate methods.</td>
</tr>
<tr>
<td><strong>RunNotifier</strong></td>
<td>If you write custom runners, you may need to notify JUnit of your progress running tests.</td>
</tr>
</tbody>
</table>

Exception Summary

<table>
<thead>
<tr>
<th>Exception Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>StoppedByUserException</strong></td>
<td>Thrown when a user has requested that the test run stop.</td>
</tr>
</tbody>
</table>
Package org.junit.runner.notification Description

Provides information about a test run.

: 4.0
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**
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:

```java
public class RingingListener extends RunListener {
    public void testFailure(Failure failure) {
        Cowbell.ring();
    }
}
```

To invoke your listener, you need to run your tests through JUnitCore.

```java
public void main(String... args) {
    JUnitCore core= new JUnitCore();
    core.addListener(new RingingListener());
    core.run(MyTestClass.class);
}
```
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

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

class 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

class testAssumptionFailure : (Failure failure) throws Exception

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

class 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
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.

```java
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.

```java
RunNotifier() { 
}

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>addFirstListener(RunListener listener)</code></td>
<td>Internal use only.</td>
</tr>
<tr>
<td><code>addListener(RunListener listener)</code></td>
<td>Internal use only</td>
</tr>
<tr>
<td><code>fireTestAssumptionFailed(Failure failure)</code></td>
<td>Invoke to tell listeners that an atomic test flagged that it assumed something false.</td>
</tr>
<tr>
<td><code>fireTestFailure(Failure failure)</code></td>
<td>Invoke to tell listeners that an atomic test failed.</td>
</tr>
<tr>
<td><code>fireTestFinished(Description description)</code></td>
<td>Invoke to tell listeners that an atomic test finished.</td>
</tr>
<tr>
<td><code>fireTestIgnored(Description description)</code></td>
<td></td>
</tr>
</tbody>
</table>
Invoke to tell listeners that an atomic test was ignored.

```java
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`

---

**RunNotifier**

public RunNotifier()  

---

**addListener**

public void addListener(RunListener listener)

    Internal use only

---

**removeListener**

public void removeListener(RunListener listener)
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

gpublic void addFirstListener(RunListener listener)

    Internal use only. The Result's listener must be first.
public class StoppedByUserException

Thrown when a user has requested that the test run stop. Writers of test running GUIs should be prepared to catch a StoppedByUserException.
StoppedByUserException

public StoppedByUserException()
org.junit.runners Class AllTests

java.lang.Object
   └ org.junit.runner.Runner
       └ org.junit.internal.runnersJUnit38ClassRunner
           └ 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.runnersJUnit38ClassRunner
AllTests

public AllTests(Class<?<(klass)
    throws Throwable

    Only called reflectively. Do not use programmatically.
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>)
protected void ComputesTestMethods()
Returns the methods that run tests.

protected List<org.junit.runners.model.FrameworkMethod> computeTestMethods()
Returns the methods that run tests.

protected Object createTest()
Returns a new fixture for running tests.

protected Description describeChild(org.junit.runners.model.FrameworkMethod method)
Returns a Description for child list returned by ParentRunner.getChildren().

protected List<org.junit.runners.model.FrameworkMethod> getChildren()
Returns a list of objects that define the children of this Runner.

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

Returns a Statement that invokes test.

Deprecated. Will be private soon.

protected List<org.junit.rules.MethodRule> rules(Object test)

protected void runChild(org.junit.runners.model.FrameworkMethod method, RunNotifier notifier)
Runs the test corresponding to child the list returned by ParentRunner.getChildren().

protected String testName(org.junit.runners.model.FrameworkMethod method)
Returns the name that describes the test.

protected void validateConstructor(List<Throwable> errors)
Adds to errors a throwable for each problem noted with the test class (available from ParentRunner.getTestClass()).

protected void validateInstanceMethods(List<Throwable> errors)
Deprecated. unused API, will go away in future version.

protected void validateOnlyOneConstructor(List<Throwable> errors)
Adds to errors if the test class has more than one constructor.

protected void validateTestMethods(List<Throwable> errors)
Adds to errors if the test class takes parameters.
Adds to errors for each method instance method with no arguments.

| protected void validateZeroArgConstructor(List<Throwable>) | Adds to errors if the test class's single constructor override |

| 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**

```java
default BlockJUnit4ClassRunner(Class<? extends org.junit.After> klass)
throws org.junit.runners.model.InitializationError
```

Creates a BlockJUnit4ClassRunner to run klass
runChild

protected void runChild(org.junit.runners.model.FrameworkMethod method, 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

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> computeTestMethods

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

```java
protected String testName(org.junit.runners.model.FrameworkMethod method)

Returns the name that describes method for Description. Default implementation is the method's name
```

methodBlock

```java

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**

```java
protected org.junit.runners.model.Statement methodInvoker(org.junit. Object tes
```

Returns a Statement that invokes method on test

---

**possiblyExpectingExceptions**

```java
@Deprecated
protected org.junit.runners.model.Statement possiblyExpectingExceptions
```

**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**

```java
@Deprecated
protected org.junit.runners.model.Statement withPotentialTimeout
```

**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**

```java
@Deprecated
protected org.junit.runners.model.Statement withBefores(org.junit.runner.Description description)
```

**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**

```java
@Deprecated
```

**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**

```java
protected List<org.junit.rules.MethodRule> rules(Object test)
```

: the MethodRules that can transform the block that runs each method in the tested class.
Class 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
JUnit4

public JUnit4(Class<? extends Object> 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 org.junit.runners

Provides standard Runner implementations.

<table>
<thead>
<tr>
<th>Runner</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllTests</td>
<td>Runner for use with JUnit 3.8.x-style AllTests classes (those that only implement a static suite() method).</td>
</tr>
<tr>
<td>BlockJUnit4ClassRunner</td>
<td>Implements the JUnit 4 standard test case class model, as defined by the annotations in the org.junit package.</td>
</tr>
<tr>
<td>JUnit4</td>
<td>Aliases the current default JUnit 4 class runner, for future-proofing.</td>
</tr>
<tr>
<td>Parameterized</td>
<td>The custom runner Parameterized implements parameterized tests.</td>
</tr>
<tr>
<td>ParentRunner&lt;T&gt;</td>
<td>Provides most of the functionality specific to a Runner that implements a &quot;parent node&quot; in the test tree, with children defined by objects of some data type T.</td>
</tr>
<tr>
<td>Suite</td>
<td>Using Suite as a runner allows you to manually build a suite containing tests from many classes.</td>
</tr>
</tbody>
</table>

Annotation Types Summary

<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameterized.Parameters</td>
<td>Annotation for a method which provides parameters to be injected into the test class constructor by Parameterized.</td>
</tr>
<tr>
<td>Suite.SuiteClasses</td>
<td>The SuiteClasses annotation specifies the classes to be run when a class annotated with @RunWith(Suite.class) is run.</td>
</tr>
</tbody>
</table>
Package org.junit.runners Description

Provides standard Runner implementations.

: 4.0 :

Runner, BlockJUnit4ClassRunner
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**)
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:

```java
@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.

```java
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<? extends Object> 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**

```java
public Parameterized(Class<? extends Runnable> klass)
    throws Throwable
```

Only called reflectively. Do not use programmatically.

```java
: Throwable
```

---

**getChildren**

```java
protected List<Runner> getChildren()
```

**Description copied from class: ParentRunner**

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

```java
: getChildren in class Suite
```
<table>
<thead>
<tr>
<th>Element</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REQUIRED</td>
</tr>
</tbody>
</table>

...
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
org.junit.runners  **Class ParentRunner<T>**

```java
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.

---

```java
protected ParentRunner(Class<? extends T> testClass)

Constructs a new ParentRunner that will run @TestClass
```

---

```java
protected org.junit.runners.model.Statement

childrenInvoker(RunNotifier notifier)

Returns a Statement: Call runChild(Objec

each object returned by getChildren() (subject to sort)
```
<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>protected</td>
<td><code>org.junit.runners.model.Statement</code></td>
<td><code>classBlock(RunNotifier notifier)</code> Constructs a Statement to run all of the tests.</td>
</tr>
<tr>
<td>protected void</td>
<td></td>
<td><code>collectInitializationErrors(List&lt;Throwable&gt; errors)</code> Adds to errors a throwable for each problem with the test class (available from <code>getTestClass()</code>).</td>
</tr>
<tr>
<td>protected abstract</td>
<td><code>Description</code></td>
<td><code>describeChild(T child)</code> Returns a <code>Description</code> for child, which can be an element of the list returned by <code>getChildren()</code>.</td>
</tr>
<tr>
<td>void</td>
<td><code>Filter filter</code></td>
<td><code>filter(Filter filter)</code> Remove tests that don't pass the parameter.</td>
</tr>
<tr>
<td>protected abstract</td>
<td><code>List&lt;T&gt;</code></td>
<td><code>getChildren()</code> Returns a list of objects that define the children.</td>
</tr>
<tr>
<td></td>
<td><code>Description</code></td>
<td><code>getDescription()</code></td>
</tr>
<tr>
<td>protected</td>
<td><code>String</code></td>
<td><code>getName()</code> Returns a name used to describe this Runner.</td>
</tr>
<tr>
<td>org.junit.runners.model.TestClass</td>
<td></td>
<td><code>getTestClass()</code> Returns a TestClass object wrapping the class.</td>
</tr>
<tr>
<td>void</td>
<td><code>RunNotifier notifier</code></td>
<td><code>run(RunNotifier notifier)</code> Run the tests for this runner.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>runChild(T child, RunNotifier notifier)</code> Runs the test corresponding to child, which can be an element of the list returned by <code>getChildren()</code>.</td>
</tr>
<tr>
<td>void</td>
<td><code>Sorter sorter</code></td>
<td><code>sort(Sorter sorter)</code> Sorts the tests using sorter.</td>
</tr>
<tr>
<td>protected void</td>
<td><code>Class&lt;? extends Annotation&gt; annotation, boolean isStatic, List&lt;Throwable&gt; errors</code></td>
<td><code>validatePublicVoidNoArgMethods(Class&lt;? extends Annotation&gt; annotation, boolean isStatic, List&lt;Throwable&gt; errors)</code> Adds to errors if any method in this class annotation, but: is not public, or takes parameters other than void, or is static (given <code>isStatic</code> is false) (given <code>isStatic</code> is true).</td>
</tr>
<tr>
<td></td>
<td><code>org.junit.runners.model.Statement</code></td>
<td><code>withAfterClasses(org.junit.runners.model.Statement)</code></td>
</tr>
</tbody>
</table>
Returns a Statement: run all non-overridden methods on this class and superclasses before executing statement methods are always executed: exceptions thrown combined, if necessary, with exceptions from AfterClass methods into a MultipleFailureException.

withBeforeClasses(org.junit.runners.model.Statement statement)

Returns a Statement: run all non-overridden on this class and superclasses before executing an Exception, stop execution and pass the exception on.

ParentRunner

protected ParentRunner(Class<? extends ParentRunner> 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(T 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(T 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.junit.runners.model.Statement s)

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.junit.runners.model.Statement s)

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

```java
public void setScheduler(org.junit.runners.model.RunnerScheduler scheduler)
```

Sets a scheduler that determines the order and parallelization of children. Highly experimental feature that may change.
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 Test 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.

---

<table>
<thead>
<tr>
<th>static interface</th>
<th>Suite.SuiteClasses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The SuiteClasses annotation specifies the classes to be run when a class annotated with @RunWith(Suite.class) is run.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>protected Suite(Class&lt;?&gt; klass, Class&lt;?&gt;[] suiteClasses)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call this when the default builder is good enough.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>protected Suite(Class&lt;?&gt; klass, List&lt;Runner&gt; runners)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Called by this class and subclasses once the runners making up the suite have been determined</td>
</tr>
</tbody>
</table>

<p>| Suite(Class&lt;?&gt; klass, |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Suite(org.junit.runners.model.RunnerBuilder builder, Class&lt;?&gt;[] classes)</code></td>
<td>Call this when there is no single root class (for example, multiple class names passed on the command line to JUnitCore).</td>
</tr>
<tr>
<td><code>protected Suite(org.junit.runners.model.RunnerBuilder builder, Class&lt;?&gt; klass, Class&lt;?&gt;[] suiteClasses)</code></td>
<td>Called by this class and subclasses once the classes making up the suite have been determined.</td>
</tr>
<tr>
<td><code>protected Description describeChild(Runner child)</code></td>
<td>Returns a <code>Description</code> for child, which can be assumed to be an element of the list returned by <code>ParentRunner.getChildren()</code>.</td>
</tr>
<tr>
<td><code>static Runner emptySuite()</code></td>
<td>Returns an empty suite.</td>
</tr>
<tr>
<td><code>protected List&lt;Runner&gt; getChildren()</code></td>
<td>Returns a list of objects that define the children of this Runner.</td>
</tr>
<tr>
<td><code>protected void runChild(Runner runner, RunNotifier notifier)</code></td>
<td>Runs the test corresponding to child, which can be assumed to be an element of the list returned by <code>ParentRunner.getChildren()</code>.</td>
</tr>
</tbody>
</table>

**ParentRunner**

- `childrenInvoker`, `classBlock`, `collectInitializationErrors`, `filter`, `getDescription`, `getName`, `getTestClass`, `run`, `setScheduler`, `sort`, `validatePublicVoidNoArgMethods`, `withAfterClasses`, `withBeforeClasses`
Suite


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


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

### Required Element Summary

<table>
<thead>
<tr>
<th>Class&lt;&lt;?&gt;[] value</th>
</tr>
</thead>
</table>

- **value**
  
  public abstract `Class<>[] value`

  : the classes to be run
After
AfterClass
AllOf
AllTests
AnyOf
Assert
Assume
Before
BeforeClass
BlockJUnit4ClassRunner
ComparisonFailure
Computer
Describable
DescribedAs
Description
Failure
Filter
Filterable
Ignore
Is
IsAnything
IsEqual
IsInstanceOf
Isn't
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