

Junit report



By: ahmed kamel taha

# What is Junit?

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit that originated with JUnit.

JUnit promotes the idea of "first testing then coding", which emphasis on setting up the test data for a piece of code which can be tested first and then can be implemented . This approach is like "test a little, code a little, test a little, code a little..." which increases programmer productivity and stability of program code that reduces programmer stress and the time spent on debugging.

# Features

JUnit is an open source framework which is used for writing & running tests.

Provides Annotation to identify the test methods.

Provides Assertions for testing expected results.

Provides Test runners for running tests.

JUnit tests allow you to write code faster which increasing quality

JUnit is elegantly simple. It is less complex & takes less time.

JUnit tests can be run automatically and they check their own results and provide immediate feedback. There's no need to manually comb through a report of test results.

JUnit tests can be organized into test suites containing test cases and even other test suites.

Junit shows test progress in a bar that is green if test is going fine and it turns red when a test fails.

# What is a Unit Test Case ?

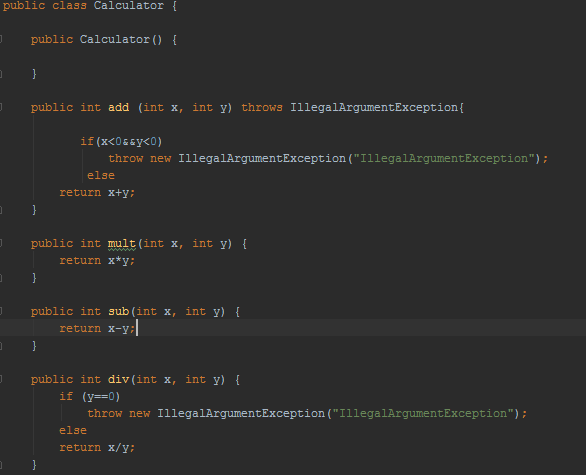
A Unit Test Case is a part of code which ensures that the another part of code (method) works as expected. To achieve those desired results quickly, test framework is required .JUnit is perfect unit test framework for java programming language.

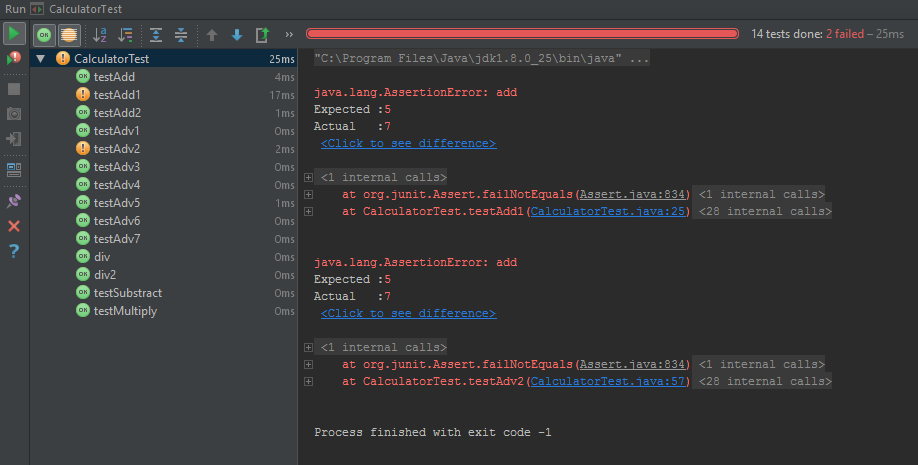
A formal written unit test case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input should test a precondition and the expected output should test a post condition.

There must be at least two unit test cases for each requirement: one positive test and one negative test. If a requirement has sub-requirements, each sub-requirement must have at least two test cases as positive and negative.

|  |  |
| --- | --- |
| Manual testing | Automated testing |
| Executing the test cases manually without any tool support is known as manual testing.  Time consuming and tedious: Since test cases are executed by human resources so it is very slow and tedious.  Huge investment in human resources: As test cases need to be executed manually so more testers are required in manual testing.  Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors.  Non-programmable: No programming can be done to write sophisticated tests which fetch hidden information. | Taking tool support and executing the test cases by using automation tool is known as automation testing.  Fast Automation runs test cases significantly faster than human resources.  Less investment in human resources: Test cases are executed by using automation tool so less tester are required in automation testing.  More reliable: Automation tests perform precisely same operation each time they are run.  Programmable: Testers can program sophisticated tests to bring out hidden information. |

# Screenshots from the project





# Using junit with intellij

# **Run/Debug Configuration: JUnit**

JUnit run/debug configurations define how unit tests that are based on the JUnit testing framework should be run.

The dialog box consists of the following tabs:

* [Configuration tab](https://www.jetbrains.com/idea/help/run-debug-configuration-junit.html#configTab)
* [Code Coverage tab](https://www.jetbrains.com/idea/help/run-debug-configuration-junit.html#codeCoverageTab)
* [Logs tab](https://www.jetbrains.com/idea/help/run-debug-configuration-junit.html#logsTab)

## **Configuration tab**

| Item | Description | |
| --- | --- | --- |
| Test kind | From this drop-down list, select the scope for your tests and fill in the fields depending on your selection. | |
| All in package | Select this option to run all unit tests in the specified package. Fill in the following fields: | |
| Package | Specify package name |
| Search for tests | Select where in your project IntelliJ IDEA shall look for test classes related to the current package:   * In whole project: IntelliJ IDEA will look for test classes in all project modules * In single module: IntelliJ IDEA will look for test classes only in the module selected in theUse classpath of module field * Across multiple dependencies: IntelliJ IDEA will look for test classes only in the module selected in the Use classpath of module field, and in the modules that depend on it |
| All in directory | Select this option to run all unit tests in the specified directory. Fill in the following field: | |
| Directory | Specify the directory where you want to run the tests. It will act as the root directory for all relative input and output paths. |
| Pattern | Select this option to run a set of test classes. This set may include classes located in the same or different directories, packages or modules. Fill in the following fields: | |
| Pattern | Specify the required classes. Each class in this field must be represented by its fully qualified name. Class names must be separated with ||. You can type class names manually, or clickadd on the right (or press Shift+Enter) and search for classes you want to add in the dialog that opens.  You can also create a suite test, i.e. a bundle of several test classes that will be run together. To create a suite test class, click edit_scopes_icon on the right and type the test classes you want to be run as a suite in the Configure suit tests dialog that opens. As a result, a new class will be created with the @Suite annotation. |
| Method | Specify the method to be launched (passed to the JRE). Type method name, or click browseButton and select the desired method in the dialog that opens. |
| Search for tests | Select where in your project IntelliJ IDEA shall look for test classes related to the current package:   * In whole project: IntelliJ IDEA will look for test classes in all project modules * In single module: IntelliJ IDEA will look for test classes only in the module selected in theUse classpath of module field * Across multiple dependencies: IntelliJ IDEA will look for test classes only in the module selected in the Use classpath of module field, and in the modules that depend on it |
| Class | Select this option to run all tests in a class.  Fill in the following field: | |
| Class | Specify the fully qualified name of the class to be launched (passed to the JRE). Type the class name or click browseButton and select the desired class in the dialog that opens. |
| Method | Select this option to run an individual test method.  Fill in the following fields: | |
| Class | Specify the fully qualified name of the class to be launched (passed to the JRE). Type the class name or click browseButton and select the desired class in the dialog that opens. |
| Method | Specify the method to be launched (passed to the JRE). Type method name, or click browseButton and select the desired method in the dialog that opens. |
| Category | Select this option if you only want to run test classes and test methods that are annotated either with the category given with the @IncludeCategory annotation, or a subtype of this category. [Learn more about JUnit categories](https://github.com/junit-team/junit/wiki/Categories).  Fill in the following fields: | |
| Category | Specify the desired category. Type category name, or click browseButton and select the desired category in the dialog that opens. |
| Search for tests | Select where in your project IntelliJ IDEA shall look for test classes related to the current package:   * In whole project: IntelliJ IDEA will look for test classes in all project modules * In single module: IntelliJ IDEA will look for test classes only in the module selected in theUse classpath of module field * Across multiple dependencies: IntelliJ IDEA will look for test classes only in the module selected in the Use classpath of module field, and in the modules that depend on it |
| Fork mode | This option controls how many Java VMs will be created if you want to fork some tests. Select method orclass to create a separate virtual machine for each method or class respectively.  The available options in this drop-down list depend on the [Test kind](https://www.jetbrains.com/idea/help/run-debug-configuration-junit.html#testKind) setting. | |
| Repeat | If you want to repeatedly run a test, select the threshold from this drop-down list. You can select to run your test once, n times (in this case specify the number of times in the field on the right), until the test fails, or until it is stopped. | |
| VM options | If necessary, specify the string to be passed to the VM. This string may contain the options such as -mx,-verbose, etc.  When specifying the options, follow these rules:   * Use spaces to separate individual options, for example, -client -ea -Xmx1024m. * If an option includes spaces, enclose the spaces or the argument that contains the spaces in double quotes, for example, some" "arg or "some arg". * If an option includes double quotes (e.g. as part of the argument), escape the double quotes by means of the backslashes, for example, -Dmy.prop=\"quoted\_value\".   If there is not enough space, you can click edit_scopes_icon and enter the string in the dialog that opens.  The -classpath option specified in this field overrides the classpath of the module. | |
| Program arguments | In this field, type a list of arguments to be passed to the program in the format you would use in the command line. If necessary, click the edit_scopes_icon button and type the required arguments in the dialog that opens.  Use the same rules as for specifying the [VM options](https://www.jetbrains.com/idea/help/run-debug-configuration-junit.html#vm_options). | |
| Working directory | Specify the directory that will act as the current directory when running the test. It will act as the root directory for all relative input and output paths. By default, the directory where the project file resides, is used as a working directory.  Type directory name, or click browseButton and select the desired directory in the dialog that opens. You can also click icon_switch_directories to switch between directories. | |
| Environment variables | Click browseButton to open the Environment Variables dialog box where you can create variables and specify their values. | |
| Use classpath of module | From this drop-down list, select the module whose classpath will be used to run the application. | |
| Use alternative JRE | Select this option to use a JRE that is different from the one associated with your project or module. Choose a JRE from the drop-down list, or click the browseButton button and browse to the required JRE in the dialog that opens. | |

## **Code Coverage tab**

Use this tab to configure [code coverage](https://www.jetbrains.com/idea/help/code-coverage.html) monitoring options.

| Item | Description |
| --- | --- |
| Choose code coverage runner | Select the desired code coverage runner.  By default, IntelliJ IDEA uses its own coverage engine with the Sampling mode. You can also choose [JaCoCo](http://www.eclemma.org/jacoco/) or [Emma](http://emma.sourceforge.net/) for calculating coverage. |
| Sampling | Select this option to measure code coverage with minimal slow-down. |
| Tracing | Select this option to collect accurate branch coverage. This mode is available for the IntelliJ IDEA code coverage runner only. |
| Track per test coverage | Select this check box to detect lines covered by one test and all tests covering line. If this check box is selected, junitIcon becomes available on the toolbar of the coverage statistic pop-up window.  This option is only available for the Tracing mode of code coverage measurement for the testing run/debug configurations.  Refer to the section [Viewing Code Coverage Results](https://www.jetbrains.com/idea/help/viewing-code-coverage-results.html#junit). |
| Merge data with previous results | When you run your unit testing or application configuration several times, use this item to calculate statistics in the Project View, taking into account the statistics of each time you have run the configuration.  Finally, the line is considered *covered* if it is covered at least once. |
| Packages and classes to record code coverage data | Click add-class and add-package buttons to specify classes and packages to be measured. You can also remove classes and packages from the list by selecting them in the list and clicking the delete button. |
| Enable coverage in test folders. | If this check box is selected, the folders marked as test rootTest are included in the code coverage analysis. |

## **Logs tab**

Use this tab to specify which log files generated while running or debugging should be displayed in the console, that is, on the dedicated tabs of the [Run](https://www.jetbrains.com/idea/help/run-tool-window.html) or [Debug tool window](https://www.jetbrains.com/idea/help/debug-tool-window.html).

| Item | Description |
| --- | --- |
| Is Active | Select check boxes in this column to have the log entries displayed in the corresponding tabs in the [Run tool window](https://www.jetbrains.com/idea/help/run-tool-window.html) or [Debug tool window](https://www.jetbrains.com/idea/help/debug-tool-window.html). |
| Log File Entry | The read-only fields in this column list the log files to show. The list can contain:   * Full paths to specific files. * [Ant patterns](http://ant.apache.org/manual/dirtasks.html#patterns) that define the range of files to be displayed. * Aliases to substitute for full paths or patterns. These aliases are also displayed in the headers of the tabs where the corresponding log files are shown.   If a log entry pattern defines more than one file, the tab header shows the name of the file instead of the log entry alias. |
| Skip Content | Select this check box to have the previous content of the selected log skipped. |
| Save console output to file | Select this check box to save the console output to the specified location. Type the path manually, or click the browse button and point to the desired location in the [dialog that opens](https://www.jetbrains.com/idea/help/select-path-dialog.html). |
| Show console when a message is printed to standard output stream | Select this check box to activate the output console and bring it forward if an associated process writes to Standard.out. |
| Show console when a message is printed to standard error stream | Select this check box to activate the output console and bring it forward if an associated process writes to Standard.err. |
| add | Click this button to open the [Edit Log Files Aliases dialog](https://www.jetbrains.com/idea/help/edit-log-files-aliases-dialog.html) where you can select a new log entry and specify an alias for it. |
| edit1 | Click this button to edit the properties of the selected log file entry in the [Edit Log Files Aliases dialog](https://www.jetbrains.com/idea/help/edit-log-files-aliases-dialog.html). |
| delete | Click this button to remove the selected log entry from the list. |
| browseButton.png | Click this button to edit the select log file entry. The button is available only when an entry is selected. |

## **Toolbar**

| Item | Shortcut | Description |
| --- | --- | --- |
| add | Alt+Insert | Click this button to add a new configuration to the list. |
| delete | Alt+Delete | Click this button to remove the selected configuration from the list. |
| copy | Ctrl+D | Click this button to create a copy of the selected configuration. |
| settings | Edit defaults | Click this button to edit the default configuration templates. The defaults are used for newly created configurations. |
| arrowUp or arrowDown | Alt+Up or Alt+Down | Use these buttons to move the selected configuration or folder up and down in the list.  The order of configurations or folders in the list defines the order in which configurations appear in the Run/Debug drop-down list on the main toolbar. |
| folder | Move into new folder / Create new folder | Use this button to [create a new folder](https://www.jetbrains.com/idea/help/creating-folders-and-grouping-run-debug-configurations.html).  If one or more run/debug configurations are in focus, the selected run/debug configurations are automatically moved to the newly created folder. If only a category is in focus, an empty folder is created.  Move run/debug configurations to a folder using drag-and-drop, or the arrowUp arrowDown buttons. |
| sortAlphabetically | Sort configurations | Click this button to sort configurations in alphabetical order. |

## **Common options**

| Item | Description |
| --- | --- |
| Name | In this text box, specify the name of the current run/debug configuration. This field does not appear for the default run/debug configurations. |
| Defaults | This node in the left-hand pane of the dialog box contains the default run/debug configuration settings. Select the desired configuration to change its default settings in the right-hand pane. The defaults are applied to all newly created run/debug configurations. |
| Share | Select this check box to make the run/debug configuration available to other team members.  If the [directory-based project format](https://www.jetbrains.com/idea/help/project.html#newprojectformat) is used, the settings for a run/debug configuration are stored in a separate .xml file in the .idea\runConfigurations folder if the run/debug configuration is shared, or in the .idea\workspace.xml file otherwise.  If the [file-based format](https://www.jetbrains.com/idea/help/project.html#file_based_format) is used, the settings are stored in the .ipr file for shared configurations, or in the .iws file otherwise.  This check box is not available when editing the run/debug configuration defaults. |
| Before launch | Specify which tasks must be performed before applying the run/debug configuration. The specified tasks are performed in the order they appear in the list.   | Item | Keyboard shortcut | Description | | --- | --- | --- | | add | Alt+Insert | Click this icon to add a task to the list. Select the task to be added:   * Run External tool. Select this option to run an application which is external to IntelliJ IDEA. In the dialog that opens, select the application or applications that should be run. If the necessary application is not defined in IntelliJ IDEA yet, add its definition. For more information, see [Configuring Third-Party Tools](https://www.jetbrains.com/idea/help/configuring-third-party-tools.html) and [External Tools](https://www.jetbrains.com/idea/help/external-tools.html). * Make. Select this option to have the project or module compiled. The[Make Module command](https://www.jetbrains.com/idea/help/compilation-types.html#make_module) will be carried out if a particular module is specified in the run/debug configuration, and the [Make Project command](https://www.jetbrains.com/idea/help/compilation-types.html#make_project)otherwise.   If an error occurs during the compilation, IntelliJ IDEA won't attempt to start the run/debug configuration.   * Make, no error check. The same as the Make option but IntelliJ IDEA will try to start the run/debug configuration irrespective of the compilation result. * Build Artifacts. Select this option to have an [artifact](https://www.jetbrains.com/idea/help/artifact.html) or artifacts built. In the dialog that opens, select the artifact or artifacts that should be built.   See also, [Configuring Artifacts](https://www.jetbrains.com/idea/help/configuring-artifacts.html).   * Run Ant target. Select this option to have an Ant target run. In the dialog that opens, select the target to be run. For more information, see [Ant](https://www.jetbrains.com/idea/help/ant.html). * Generate CoffeeScript Source Maps. Select this option to have the source maps for your CoffeeScript sources generated. In the dialog that opens, specify where your CoffeeScript source files are located. For more information, see [CoffeeScript Support](https://www.jetbrains.com/idea/help/coffeescript-support.html). * Run Maven Goal. Select this option to have a Maven goal run. In the dialog that opens, select the goal to be run.   For more information, see [Maven](https://www.jetbrains.com/idea/help/maven.html).   * Run Remote External tool: Add a remote SSH external tool. Refer to the section [Remote SSH External Tools](https://www.jetbrains.com/idea/help/remote-ssh-external-tools.html) for details. | | delete | Alt+Delete | Click this icon to remove the selected task from the list. | | edit1 | Enter | Click this icon to edit the selected task. Make the necessary changes in the dialog that opens. | | arrowUp | Alt+Up | Click this icon to move the selected task one line up in the list. | | arrowDown | Alt+Down | Click this icon to move the selected task one line down in the list. | | Show this page |  | Select this check box to have the run/debug configuration settings shown prior to actually starting the run/debug configuration. | | Active tool window |  | Select this option if you want the [Run](https://www.jetbrains.com/idea/help/run-tool-window.html)/[Debug](https://www.jetbrains.com/idea/help/debug-tool-window.html) tool windows to be activated automatically when you run/debug your application. This option is enabled by default. | |