|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Date** | **Auteur** | **Description** |
| 001 | 05/12/16 | Gregory Amirthanathan | First version  Overview of Junit |
| 002 | 06/12/16 | Gregory Amirthanathan | Second version  Installation of Junit  Set up Junit Test for Java Program |
| 003 | 00/00/00 | Gregory Amirthanathan | Third version  Organise JUnit tests in project |
|  |  |  |  |

* What is Junit?
* JUnit is a unit testing framework for Java programming language.

## Features of Junit

* Junit is an open source framework, which is used for writing and running tests.
* Provides annotations to identify test methods.
* Provides assertions for testing expected results.
* Provides test runners for running tests.
* JUnit tests allow you to write codes faster, which increase quality.
* JUnit is elegantly simple. It is less complex and 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 the test is running smoothly, and it turns red when a test fails.

## JUnit Framework can be easily integrated with either of the following −

* Eclipse
* Ant
* Maven

# Download and Install

To download and install JUnit you currently have the following options.

# Plain-old JAR

Download the following JARs put them on your test classpath:

* [junit.jar](http://bit.ly/My9IXz)
* [hamcrest-core.jar](http://bit.ly/1gbl25b)

# Maven

Add a dependency to junit:junit in test scope. (Note: 4.12 is the latest stable version as of the latest edit on this page.)

<dependency>

<groupId>junit</groupId>

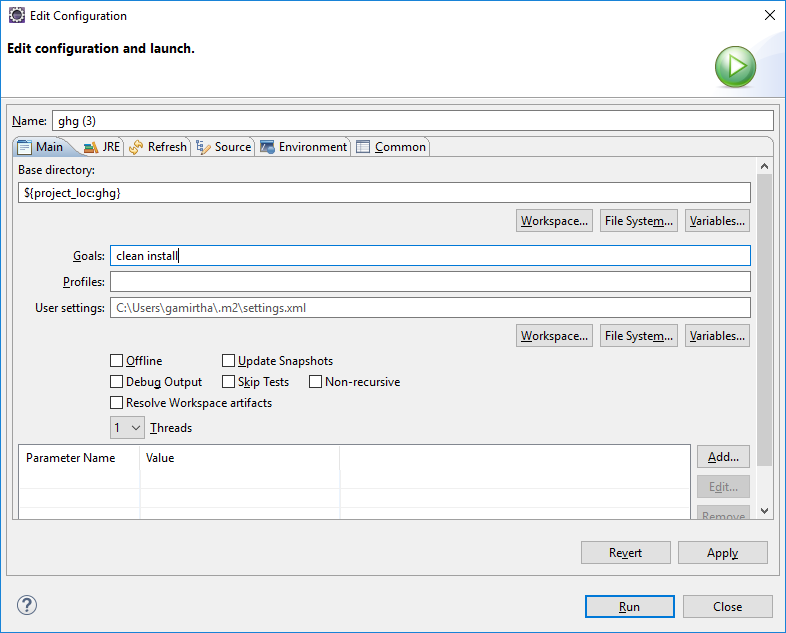
<artifactId>junit</artifactId>

<version>4.12</version>

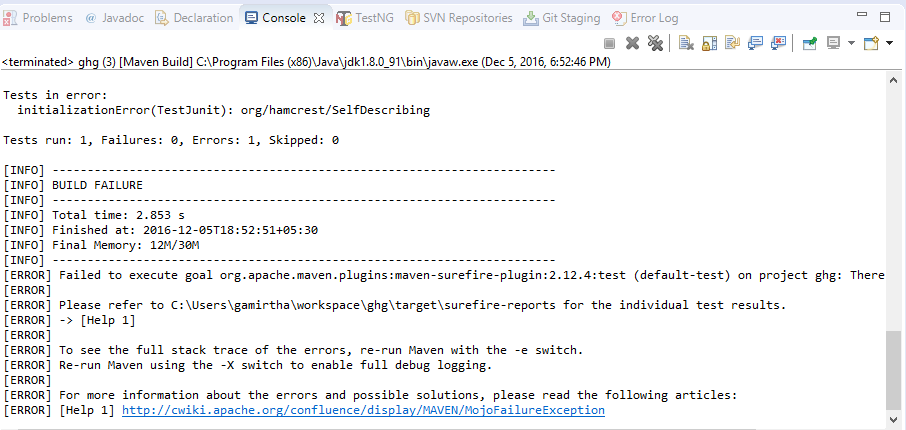
<scope>test</scope>

</dependency>

Run Maven build,



And see build status,



Note – Above status is failed, however adding library below helps.

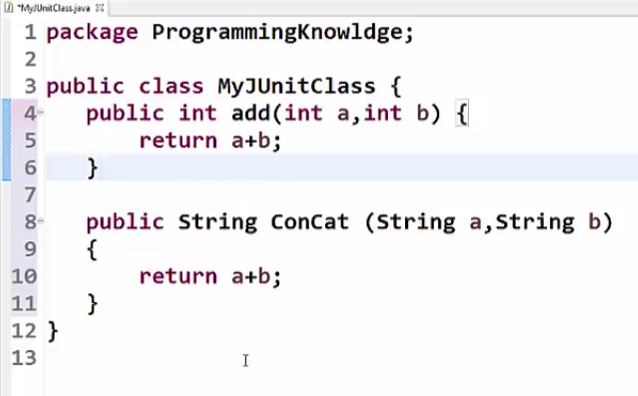
## ADD Junit Library to Eclipse

A few steps you have to follow:

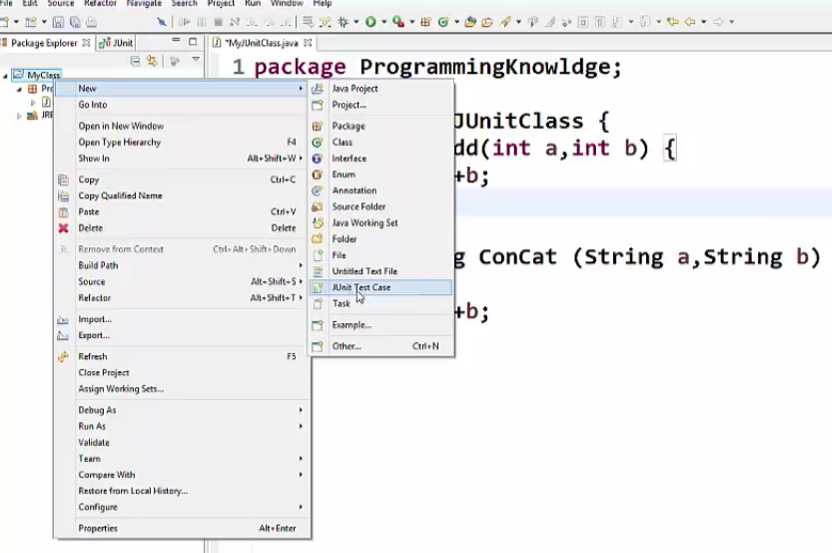
1. Right click on the project.
2. Choose Build Path Then from its menu choose Add Libraries.
3. Choose JUnit then click Next.
4. Choose JUnit4 then Finish.

Set up Junit Test for Java Program

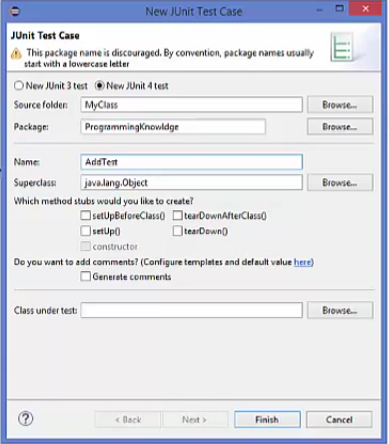
1. Create New Java Project | New package | New Class

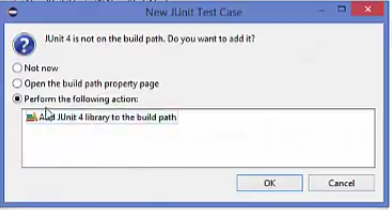


Right click on project 🡪 select ‘New | Junit Test Case’



Give Name | finish | ok

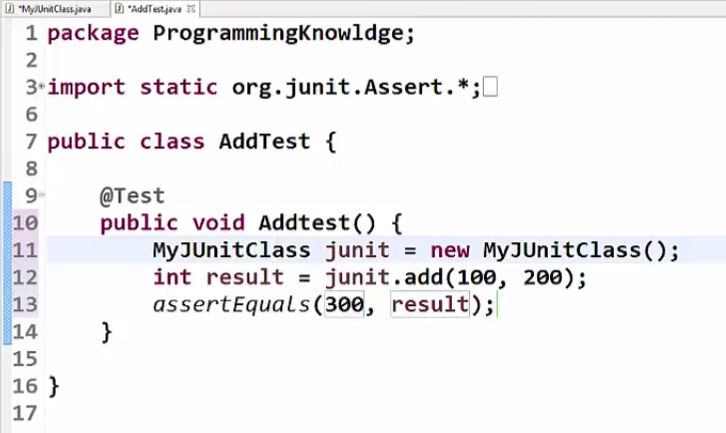




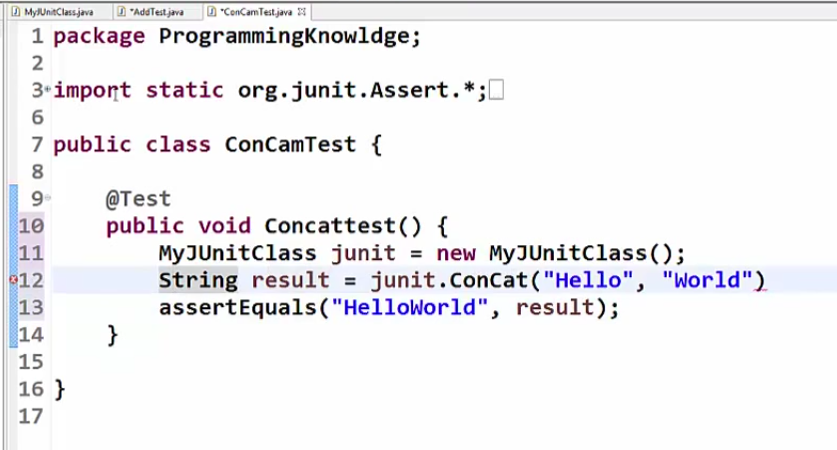
Create object of java class

Use the function to be tested

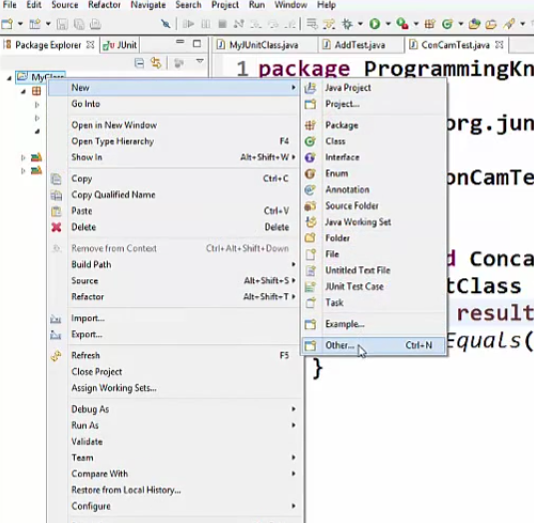
Use Assertions

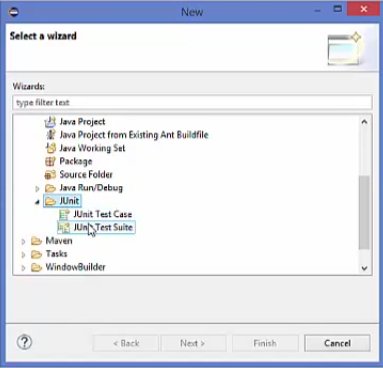


New Java Class



Create test Suite

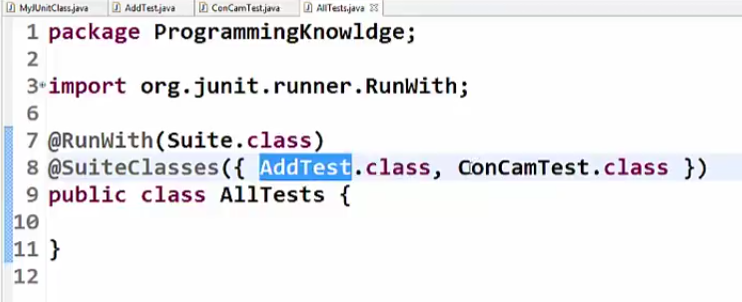




Next | Add package | Add tests | finish



See the test suite



## Test JUnit Setup

Create a java class file name TestJunit

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class TestJunit {

@Test

public void testAdd() {

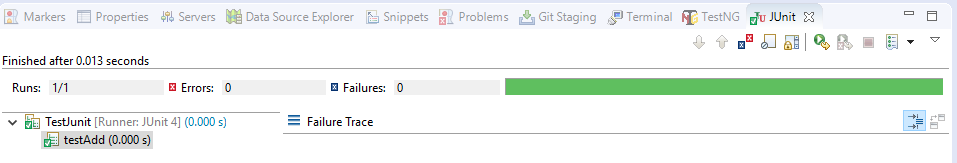
String str = "Junit is working fine";

assertEquals("Junit is working fine",str);

}

}

Execution status –



Create a java class file name TestRunner

import org.junit.runner.JUnitCore;

import org.junit.runner.Result;

import org.junit.runner.notification.Failure;

public class TestRunner {

public static void main(String[] args) {

Result result = JUnitCore.runClasses(TestJunit.class);

for (Failure failure : result.getFailures()) {

System.out.println(failure.toString());

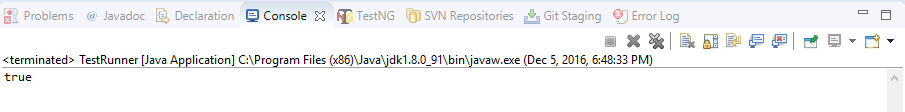
}

System.out.println(result.wasSuccessful());

}

}

Execution status –



* Junit Assertions

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | **void assertEquals(boolean expected, boolean actual)**  Checks that two primitives/objects are equal. |
| 2 | **void assertTrue(boolean expected, boolean actual)**  Checks that a condition is true. |
| 3 | **void assertFalse(boolean condition)**  Checks that a condition is false. |
| 4 | **void assertNotNull(Object object)**  Checks that an object isn't null. |
| 5 | **void assertNull(Object object)**  Checks that an object is null. |
| 6 | **void assertSame(boolean condition)**  The assertSame() method tests if two object references point to the same object. |
| 7 | **void assertNotSame(boolean condition)**  The assertNotSame() method tests if two object references do not point to the same object. |
| 8 | **void assertArrayEquals(expectedArray, resultArray);**  The assertArrayEquals() method will test whether two arrays are equal to each other. |

* Annotations
* The beforeClass() method executes only once.
* The afterClass() method executes only once.
* The before() method executes for each test case, but before executing the test case.
* The after() method executes for each test case, but after the execution of test case.
* In between before() and after(), each test case executes.
* @RunWith and @Suite annotations are used to run the suite tests
* A test method annotated with @Ignore will not be executed.
* The timeout parameter is used along with @Test(timeout) annotation
* The **expected** parameter is used along with @Test(expected) annotation
* Annotate test class with @RunWith(Parameterized.class).
* Create a public static method annotated with @Parameters that returns a Collection of Objects (as Array) as test data set.
* Create a public constructor that takes in what is equivalent to one "row" of test data.
* Create an instance variable for each "column" of test data.
* Create your test case(s) using the instance variables as the source of the test data.