JUNIT FAQ’s:

**1) Explain what is JUnit?**

JUnit is a testing framework for unit testing.  It uses Java as a programming platform, and it is an Open Source Software managed by the JUnit.org community.

**2) Explain what is Unit Test Case?**

Unit Test Case is a part of the code that ensures that the another part of code (method) behaves as expected. For each requirement, there must be at least two test cases one negative test and one positive test.

**3) Explain how you can write a simple JUnit test case?**

* Determine a subclass of TestCase
* To initialize object(s) under test, override the setup() method
* To release object(s) under test override the teardown() method

Determine one or more public test XYZ() methods that exercise the objects under test and assert expected results.

**4) Mention what are parameterized tests?**

Parameterized tests enable developer to perform the same test over and again using different values.

**5) Mention what is the difference between JUnit and TestNG?**

|  |  |
| --- | --- |
| **JUnit** | **TestNG** |
| * In JUnit the naming convention for annotation is a bit complicated for, e.g., “Before”, “After” and “Expected” * In JUnit, for a method declaration you have to follow a specific style like using “@BeforeClass” and “@AfterClass”. * In JUnit method name constraint is present * JUnit framework does not have “Parameterized Test” or “Dependency Test” feature * In JUnit grouping of test cases are not available * JUnit does not support parallel execution on Selenium test cases * It cannot re-run the failed cases | * In TestNG it is easier to understand annotations like “BeforMethod”, “AfterMethod” and “ExpectedException” * In TestNG, there is no restriction like you have to declare methods in a specific format * In TestNG method name constraint is not present, and you can determine any test method names * TestNG use “dependOnMethods” to implement the dependency testing * In TestNG, grouping of test cases is available * In TestNG Parallel execution of Selenium test cases are possible * It can rerun the failed tests |

**6) Mention different methods of exception handling in JUnit?**

There are different methods of exception handling in JUnit

* Try catch idiom
* With JUnit rule
* With @Test annotation
* With catch exception library
* With customs annotation

**7) Explain what is ignore test in JUnit?**

When your code is not ready, and it would fail if executed then you can use **@Ignore** annotation.

* It will not execute a test method annotated with **@Ignore**
* It will not execute any of the test methods of test class if it is annotated with **@Ignore**

**8) List out some useful JUnit extensions?**

JUnit extensions include

* Cactus
* JWebUnit
* XMLUnit
* MockObject

**9) Explain who should use JUnit – a developer or tester? Why you use JUnit to test your code?**

JUnit is more often used by developers to implement unit tests in JAVA.  It is designed for unit testing that is more a coding process and not a testing process. However, many testers and QA engineers use JUnit for unit testing.

JUnit is used because

* It test early and does automate testing
* JUnit tests can be compiled with the build so that at unit level, regression testing can be done
* It allows test code re-usage
* JUnit tests behave as a document for the unit tests when there is a transfer

**10) Explain what is JUnitCore Class?**

JUnitCore class is an inbuilt class in JUnit package; it is based on Façade design pattern, this class is used to run only definite test classes only.

**11) Explain how you can run JUnit from the command window?**

To run JUnit from the command window, you have to follow the steps

* Set the CLASSPATH
* Invoke the runner:

Java org.junit.runner.JUnitCore

**Notes on Junit:**

**JUNIT**: unit testing testing framework for java. Plays a crucial role for test driven development.

Test the individual code of snippets.

For every class we write a test class and for every method we write a test method.

A framework means it’s a semi finished application, reusable code no need to repeat code

## 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 increases 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.

## What is a Unit Test Case ?

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

**More readable,more descriptive**

**Junit is backward compatible meaning it will support previous versions classes also.**

**Sample program to understand the flow of execution:**

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Ignore;

import org.junit.Test;

public class ExecutionProcedureJunit {

**//execute only once, in the starting**

@BeforeClass

public static void beforeClass() {

System.out.println("in before class");

}

**//execute only once, in the end**

public static void afterClass() {

System.out.println("in after class");

}

**//execute for each test, before executing test**

@Before

public void before() {

System.out.println("in before");

}

**//execute for each test, after executing test**

@After

public void after() {

System.out.println("in after");

}

**//test case 1**

@Test

public void testCase1() {

System.out.println("in test case 1");

}

**//test case 2**

@Test

public void testCase2() {

System.out.println("in test case 2");

}

}

See the above output. The execution procedure is as follows −

* First of all, 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.

**Differences between junit3 and junit4:**

Annotations are used as it makes code more readable and descriptive

1. **No need to extend testcase class**
2. **We have a static import of assert statements.(we are no more depending on testcase superclass for using assert methods)**
3. **We need not append Test word before every method—we need to just use @Test annotation.**
4. **We have parameter exceptedexception in Testannotation to deal with any expected exceptions so that you will not get any failure for a testcase.**
5. **@Before,@After,@BeforeClass,@AfterClass annotations help us to move any repeated or initializing logics out of each test methods. No need to use setup and tearDown.**
6. **You can provide parameters to the methods at a time instead of having different methods for different parameters ---parameterized testing.**

**You can combine the execution of two or more classes at a time using suite annotation**