JUNIT - INTERVIEW QUESTIONS

Dear readers, these **JUnit Interview Questions** have been designed specially to get you acquainted with the nature of questions you may encounter during your interview for the subject of **JUnit**. As per my experience good interviewers hardly plan to ask any particular question during your interview, normally questions start with some basic concept of the subject and later they continue based on further discussion and what you answer:

What is Testing?

Testing is the process of checking the functionality of the application whether it is working as per requirements.

What is Unit Testing?

Unit testing is the testing of single entity *classormethod*. Unit testing is very essential to every software company to give a quality product to their customers.

What is Manual testing?

Executing the test cases manually without any tool support is known as manual testing.

What is Automated testing?

Taking tool support and executing the test cases by using automation tool is known as automation testing.

What are the disadvantages of manual testing?

Following are the disadvantages of 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.

What are the advantages of automated testing?

Following are the advantages of automated 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.

What is JUnit?

JUnit is a Regression Testing Framework used by developers to implement unit testing in Java and accelerate programming speed and increase the quality of code.

What are important features of JUnit?

Following are the important features of JUnit −

- It is an open source framework.
- Provides Annotation to identify the test methods.
- Provides Assertions for testing expected results.
- Provides Test runners for running tests.
- JUnit tests can be run automatically and they check their own results and provide immediate feedback.
- 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.

What are the best practices to write a Unit Test Case?

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 postcondition.

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.

When are Unit Tests written in Development Cycle?

Tests are written before the code during development in order to help coders write the best code.

Why not just use System.out.println for testing?

Debugging the code using system.out.println will lead to manual scanning of the whole output every time the program is run to ensure the code is doing the expected operations. Moreover, in the long run, it takes lesser time to code JUnit methods and test them on class files.

How to install JUnit?

Follow the steps below−

- Download the latest version of JUnit, referred to below as junit.zip.
- Unzip the junit.zip distribution file to a directory referred to as %JUNIT_HOME%.
- Add JUnit to the classpath−

```
set CLASSPATH=%CLASSPATH%;%JUNIT_HOME%\junit.jar
```

- Test the installation by running the sample tests distributed with JUnit *sampletestsarelocatedintheinstallationdirectorydirectly, notthejunit. jarfile*. Then simply type−

```
java org.junit.runner.JUnitCore org.junit.tests.AllTests
```

- All the tests should pass with an "OK" message. If the tests don't pass, verify that junit.jar is in the CLASSPATH.

Why does JUnit only report the first failure in a single test?

Reporting multiple failures in a single test is generally a sign that the test does too much and it is too big a unit test. JUnit is designed to work best with a number of small tests. It executes each test within a separate instance of the test class. It reports failure on each test.

In Java, assert is a keyword. Won't this conflict with JUnit's assert method?

JUnit 3.7 deprecated assert and replaced it with assertTrue, which works exactly the same way. JUnit 4 is compatible with the assert keyword. If you run with the -ea JVM switch, assertions that fail will be reported by JUnit.

How do I test things that must be run in a J2EE container *e. g. servlets, EJBs*?

Refactoring J2EE components to delegate functionality to other objects that don't have to be run in a J2EE container will improve the design and testability of the software. Cactus is an open source JUnit extension that can be used for unit testing server-side java code.

Name the tools with which JUnit can be easily integrated.

JUnit Framework can be easily integrated with either of the followings −

- Eclipse
- Ant
- Maven

What are the core features of JUnit?

JUnit test framework provides following important features −

- Fixtures
- Test suites
- Test runners
- JUnit classes

What is a fixture?

Fixture is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well known and fixed environment in which tests are run so that results are repeatable. It includes following methods −

- setUp method which runs before every test invocation.
- tearDown method which runs after every test method.

What is a test suite?

Test suite means bundle a few unit test cases and run it together. In JUnit, both @RunWith and @Suite annotation are used to run the suite test.

What is a test runner?

Test runner is used for executing the test cases.

What are JUnit classes? List some of them.

JUnit classes are important classes which are used in writing and testing JUnits. Some of the important classes are−

- **Assert** − It contains a set of assert methods.
- **TestCase** − It contains a test case defines the fixture to run multiple tests.
- **TestResult** − It contains methods to collect the results of executing a test case.
- **TestSuite** − It is a Composite of Tests.

What are annotations and how are they useful in JUnit?

Annotations are like meta-tags that you can add to you code and apply them to methods or in

class. The annotation in JUnit gives us information about test methods, which methods are going to run before & after test methods, which methods run before & after all the methods, which methods or class will be ignore during execution.

How will you run JUnit from command window?

Follow the steps below−

- Set the CLASSPATH
- Invoke the runner−

```
java org.junit.runner.JUnitCore
```

What is the purpose of org.junit.Assert class?

This class provides a set of assertion methods useful for writing tests. Only failed assertions are recorded.

What is the purpose of org.junit.TestResult class?

A TestResult collects the results of executing a test case. It is an instance of the Collecting Parameter pattern. The test framework distinguishes between failures and errors. A failure is anticipated and checked for with assertions. Errors are unanticipated problems like an ArrayIndexOutOfBoundsException.

What is the purpose of org.junit.TestSuite class?

A TestSuite is a Composite of Tests. It runs a collection of test cases.

What is the purpose of @Test annotation in JUnit?

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

What is the purpose of @Before annotation in JUnit?

Several tests need similar objects created before they can run. Annotating a public void method with @Before causes that method to be run before each Test method.

What is the purpose of @After annotation in JUnit?

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.

What is the purpose of @BeforeClass annotation in JUnit?

Annotating a public static void method with @BeforeClass causes it to be run once before any of the test methods in the class.

What is the purpose of @AfterClass annotation in JUnit?

This will perform the method after all tests have finished. This can be used to perform clean-up activities.

What is @Ignore annotation and how is this useful?

Following are some of the usefulness of @Ignore annotation −

You can easily identify all @Ignore annotations in the source code, while unannotated or commented out tests are not so simple to find.

There are cases when you can't fix a code that is failing, but you still want to method to be around, precisely so that it does not get forgotten. In such cases @Ignore makes sense.

Explain the execution procedure of the JUint test API methods?

Following is how the JUnit execution procedure works−

- First of all method annotated as @BeforeClass execute only once.
- Lastly, the method annotated as @AfterClass executes only once.
- Method annotated as @Before executes for each test case but before executing the test case.
- Method annotated as @After executes for each test case but after the execution of test case.
- In between method annotated as @Before and method annotated as @After each test case executes.

What is the purpose of org.junit.JUnitCore class?

The test cases are executed using JUnitCore class. JUnitCore is a facade for running tests. It supports running JUnit 4 tests, JUnit 3.8.x tests, and mixtures.

How to simulate timeout situation in JUnit?

Junit provides a handy option of Timeout. If a test case takes more time than specified number of milliseconds then Junit will automatically mark it as failed. The timeout parameter is used along with @Test annotation.

How can you use JUnit to test that the code throws desired exception?

JUnit provides a option of tracing the Exception handling of code. You can test if a code throws desired exception or not. The expected parameter is used along with @Test annotation as follows− @Test*expected*

What are Parameterized tests?

Junit 4 has introduced a new feature Parameterized tests.Parameterized tests allow developer to run the same test over and over again using different values.

How to create Parameterized tests?

There are five steps, that you need to follow to create Parameterized tests−

- Annotate test class with @RunWith*Parameterized. class*.
- Create a public static method annotated with @Parameters that returns a Collection of Objects *asArray* 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 tests case*s* using the instance variables as the source of the test data.
- The test case will be invoked once per each row of data. Let's see Parameterized tests in action.

How do you use test fixtures?

Fixtures is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well known and fixed environment in which tests are run so that results are repeatable. It includes−

- setUp method which runs before every test invocation.
- tearDown method which runs after every test method.

How to compile a JUnit Test Class?

Compiling a JUnit test class is like compiling any other Java classes. The only thing you need watch out is that the JUnit JAR file must be included in the classpath.

What happens if a JUnit Test Method is Declared as "private"?

If a JUnit test method is declared as "private", it compiles successfully. But the execution will fail. This is because JUnit requires that all test methods must be declared as "public".

How do you test a "protected" method?

When a method is declared as "protected", it can only be accessed within the same package where the class is defined. Hence to test a "protected" method of a target class, define your test class in the same package as the target class.

How do you test a "private" method?

When a method is declared as "private", it can only be accessed within the same class. So there is no way to test a "private" method of a target class from any test class. Hence you need to perform unit testing manually. Or you have to change your method from "private" to "protected".

What happens if a JUnit test method is declared to return "String"?

If a JUnit test method is declared to return "String", the compilation will pass ok. But the execution will fail. This is because JUnit requires that all test methods must be declared to return "void".

Can you use a main Method for Unit Testing?

Yes you can test using main method. One obvious advantage seems to be that you can whitebox test the class. That is, you can test the internals of it $privatemethodsforexample$. You can't do that with unit-tests. But primarily the test framework tests the interface and the behavior from the user's perspective.

Do you need to write a test class for every class that needs to be tested?

No. We need not write an independent test class for every class that needs to be tested. If there is a small group of tests sharing a common test fixture, you may move those tests to a new test class.

When are tests garbage collected?

The test runner holds strong references to all Test instances for the duration of the test execution. This means that for a very long test run with many Test instances, none of the tests may be garbage collected until the end of the entire test run. Explicitly setting an object to null in the tearDown method, for example, allows it to be garbage collected before the end of the entire test run.

What is a Mock Object?

In a unit test, mock objects can simulate the behavior of complex, real $non-mock$ objects and are therefore useful when a real object is impractical or impossible to incorporate into a unit test.

Explain unit testing using Mock Objects.

The common coding style for testing with mock objects is to−

- Create instances of mock objects.
- Set state and expectations in the mock objects.
- Invoke domain code with mock objects as parameters.
- Verify consistency in the mock objects.

Name some of the JUnit Extensions.

Following are the JUnit extensions−

- Cactus
- JWebUnit
- XMLUnit
- MockObject

What is Cactus?

Cactus is a simple test framework for unit testing server-side java code *Servlets*, *EJBs*, *TagLibs*, *Filters*. The intent of Cactus is to lower the cost of writing tests for server-side code. It uses JUnit and extends it. Cactus implements an in-container strategy, meaning that tests are executed inside the container.

What are the core components of Cactus?

Cactus Ecosystem is made of several components−

- Cactus Framework is the heart of Cactus. It is the engine that provides the API to write Cactus tests.
- Cactus Integration Modules are front ends and frameworks that provide easy ways of using the Cactus Framework *Antscripts*, *Eclipseplugin*, *Mavenplugin*.

What is JWebUnit?

WebUnit is a Java-based testing framework for web applications. It wraps existing testing frameworks such as HtmlUnit and Selenium with a unified, simple testing interface to allow you to quickly test the correctness of your web applications.

What are the advantages of using JWebUnit?

JWebUnit provides a high-level Java API for navigating a web application combined with a set of assertions to verify the application's correctness. This includes navigation via links, form entry and submission, validation of table contents, and other typical business web application features.

The simple navigation methods and ready-to-use assertions allow for more rapid test creation than using only JUnit or HtmlUnit. And if you want to switch from HtmlUnit to other plugins such as Selenium *availablesoon*, there is no need to rewrite your tests.

What is XMLUnit?

XMLUnit provides a single JUnit extension class, XMLTestCase, and a set of supporting classes.

What is the use of supporting classes in XMLUnit?

Supporting classes allow assertions to be made about−

- The differences between two pieces of XML *viaDiffandDetailedDiffclasses*.
- The validity of a piece of XML *viaValidatorclass*.
- The outcome of transforming a piece of XML using XSLT *viaTransformclass*.
- The evaluation of an XPath expression on a piece of XML *viaclassesimplementingtheXpathEngineinterface*.
- Individual nodes in a piece of XML that are exposed by DOM Traversal *viaNodeTestclass*.

What is Next ?

Further you can go through your past assignments you have done with the subject and make sure you are able to speak confidently on them. If you are fresher then interviewer does not expect you will answer very complex questions, rather you have to make your basics concepts very strong.

Second it really doesn't matter much if you could not answer few questions but it matters that whatever you answered, you must have answered with confidence. So just feel confident during your interview. We at tutorialspoint wish you best luck to have a good interviewer and all the very best for your future endeavor. Cheers :-)