JUnit Interview Questions And Answers Guide.

Global Guideline.

https://globalguideline.com/

Global Guideline . COM

JUnit Job Interview Preparation Guide.

Question # 1

What Is JUnit?

Answer:-

What is JUnit? You need to remember the following points when answering this question:
* It is a software testing framework to for unit testing.
* It is written in Java and designed to test Java applications.
* It is an Open Source Software maintained by the JUnit.org community.
Answer from the JUnit FAQ:
JUnit is a simple, open source framework to write and run repeatable tests. It is an instance of the xUnit architecture for unit testing frameworks. JUnit features include:
* Assertions for testing expected results
* Test fixtures for sharing common test data
* Test runners for running tests
JUnit was originally written by Erich Gamma and Kent Beck.

Read More Answers.

Question # 2

Where Do You Download JUnit?

Answer:-

Where do I download JUnit? I don't think anyone will ask this question in a job interview. But the answer is simple. You should follow the download instructions from the JUnit official Website: JUnit.org.

Read More Answers.

Question # 3

Who Should Use JUnit, Developers or Testers?

Answer:-

I should say that JUnit is mostly used by developers. JUnit is designed for unit testing, which is really a coding process, not a testing process.
But many testers or QA engineers, are also required to use JUnit for unit testing. For example, I found this job title on the Internet: Lead QA Engineer - Java / J2EE / whitebox / SAP / Junit

Read More Answers.

Question # 4

Why Do You Use JUnit to Test Your Code?

Answer:-

This is a commonly asked question in a job interview. Your answer should have these points:
* I believe that writing more tests will make me more productive, not less productive.
* I believe that tests should be done as soon as possible at the code unit level.
* I believe that using JUnit makes unit testing easier and faster.

Read More Answers.

Question # 5

How Do You Install JUnit?

Answer:-

How do I install JUnit? First I will download the lastest version of JUnit.
Then I will extract all files from the downloaded file.
The most important file should be the JUnit JAR file: junit-4.4.jar, which contains all JUnit class packages.
To verify junit-4.4.jar, I will run the org.junit.runner.JUnitCore class:
java -cp junit-4.4.jar org.junit.runner.JUnitCore
JUnit version 4.4

Time: 0
OK (0 tests)

Read More Answers.

Question # 6

How To Wirte a Simple JUnit Test Class?

Answer:-

This is a common test in a job interview. You should be able to write this simple test class with one test method:

```
import org.junit.*;
public class HelloTest {
   @Test public void testHello() {
      String message = "Hello World!";
      Assert.assertEquals(12, message.length());
   }
}
```

Read More Answers.

Question # 7

How To Compile a JUnit Test Class?

Answer:-

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. For example, to compile the test class HelloTest.java described previously, you should do this:

```
javac -cp junit-4.4.jar HelloTest.java
dir HelloTest.*
  453 HelloTest.class
  183 HelloTest.java
```

The compilation is ok, if you see the HelloTest.class file.

Read More Answers.

Question # 8

How To Run a JUnit Test Class?

Answer:-

A JUnit test class usually contains a number of test methods. You can run all test methods in a JUnit test class with the JUnitCore runner class. For example, to run the test class HelloTest.java described previously, you should do this:

```
java -cp .;
junit-4.4.jar org.junit.runner.JUnitCore HelloTest
JUnit version 4.4
.
Time: 0.015
OK (1 test)
```

This output says that 1 tests performed and passed.

Read More Answers.

Question # 9

What CLASSPATH Settings Are Needed to Run JUnit?

Answer:-

It doesn't matter if you run your JUnit tests from a command line, from an IDE, or from "ant", you must define your CLASSPATH settings correctly. Here is what recommended by the JUnit FAQ with some minor changes:

To run your JUnit tests, you'll need the following elemements in your CLASSPATH:

* The JUnit JAR file.
* Location of your JUnit test classes.
* Location of classes to be tested.
* JAR files of class libraries that are required by classes to be tested.

If attempting to run your tests results in a NoClassDefFoundError, then something is missing from your CLASSPATH.

If you are running your JUnit tests from a command line on a Windows system:

```
set CLASSPATH=c:Ajunit-4.4.jar;c:Btest_classes;
  c:Btarget_classes;c:D3rd_party.jar
```

If you are running your JUnit tests from a command line on a Unix (bash) system:

```
export CLASSPATH=/A/junit-4.4.jar:/B/test_classes:
  /C/target_classes:/D/3rd_party.jar
```

Read More Answers.

Question # 10

How Do I Run JUnit Tests from Command Window?

Answer:-

To run JUnit tests from a command window, you need to check the following list:

1. Make sure that JDK is installed and the "java" command program is accessible through the PATH setting. Type "java -version" at the command prompt, you should see the JVM reports you back the version string.
2. Make sure that the CLASSPATH is defined as shown in the previous question.
3. Invoke the JUnit runner by entering the following command:

```
java org.junit.runner.JUnitCore <test class name>
```

Global Guideline . COM

Read More Answers.

Question # 11

How Do You Uninstall JUnit?

Answer:-

Uninstalling JUnit is easy. Just remember these:
* Delete the directory that contains the JUnit JAR file and other JUnit files.
* Remove the JUnit JAR file from the CLASSPATH environment variable.
* No need to stop any background processes, because JUnit does not use background process.
* No need to remove any registry settings, because JUnit does not use Windows registry.

Read More Answers.

Question # 12

How To Write a JUnit Test Method?

Answer:-

This interview question is to check if you know the basic rules about writing a JUnit test method:
* You need to mark the method as a JUnit test method with the JUnit annotation: @org.junit.Test.
* A JUnit test method must be a "public" method. This allows the runner class to access this method.
* A JUnit test method must be a "void" method. The runner class does not check any return values.
* A JUnit test should perform one JUnit assertion - calling an org.junit.Assert.assertXXX() method.

Here is a simple JUnit test method:

```
import org.junit.*;
...
   @Test public void testHello() {
      String message = "Hello World!";
      Assert.assertEquals(12, message.length());
   }
```

Read More Answers.

Question # 13

Can You Provide a List of Assertion Methods Supported by JUnit 4.4?

Answer:-

You should be able to answer this question by looking up the org.junit.Assert class API document. Here is a list of assertino methods supported by JUnit 4.4:
* assertEquals(expected, actual)
* assertEquals(message, expected, actual)
* assertEquals(expected, actual, delta)
* assertEquals(message, expected, actual, delta)
* assertFalse(condition)
* assertFalse(message, condition)
* assertNotNull(object)
* assertNotNull(message, object)
* assertNotSame(expected, actual)
* assertNotSame(message, expected, actual)
* assertNull(object)
* assertNull(message, object)
* assertSame(expected, actual)
* assertSame(message, expected, actual)
* assertTrue(condition)
* assertTrue(message, condition)
* fail()
* fail(message)
* failNotEquals(message, expected, actual)
* failNotSame(message, expected, actual)
* failSame(message)

Read More Answers.

Question # 14

Why Does Poeple Import org.junit.Assert Statically?

Answer:-

Poeple use the static import statement on org.junit.Assert to save coding time on calling its assetion methods. With a normal import statement, assertion method names must qualified with the class name like this:

```
import org.junit.Assert;
...
      Assert.assertEquals(12, message.length());
```

With a static import statement, assertion method names can be used directly like this:

```
import static org.junit.Assert.*;
...
      assertEquals(12, message.length());
```

Read More Answers.

Question # 15

What Is the @SuiteClasses Annotation?

Copyright © https://globalguideline.com

Answer:-

"@SuiteClasses" is a class annotation defined in JUnit 4.4 in org.junit.runners.Suite.SuiteClasses. It allows you to define a suite class as described in the previous question.
By the way, the API document of JUnit 4.4 has a major typo for the org.junit.runners.Suite class (Suite.html).
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.
"@SuiteClasses(TestClass1.class, ...)" should be changed to "@Suite.SuiteClasses({TestClass1.class, ...})".
Someone provided wrong information on build test suite in JUnit 4.4. Do not follow this:
JUnit provides tools to define the suite to be run and to display its results. To run tests and see the results on the console, run:

Read More Answers.

Question # 16

Why Not Just Use a Debugger for Unit Testing?

Answer:-

This is a common question in a job interview. You should answer it with these points:
* A debugger is designed for manual debugging and manual unit testing, not for automated unit testing.
* JUnit is designed for automated unit testing.
* Automated unit testing requires extra time to setup initially. But it will save your time, if your code requires changes many times in the future.

Here is how the JUnit FAQ answers this question:
Debuggers are commonly used to step through code and inspect that the variables along the way contain the expected values. But stepping through a program in a debugger is a manual process that requires tedious visual inspections. In essence, the debugging session is nothing more than a manual check of expected vs. actual results. Moreover, every time the program changes we must manually step back through the program in the debugger to ensure that nothing broke.
It generally takes less time to codify expectations in the form of an automated JUnit test that retains its value over time. If it's difficult to write a test to assert expected values, the tests may be telling you that shorter and more cohesive methods would improve your design.

Read More Answers.

Question # 17

Why Not Just Write a main() Method for Unit Testing?

Answer:-

It is possible to write a main() method in each class that need to be tested for unit testing. In the main() method, you could create test object of the class itself, and write some tests to test its methods.
However, this is not a recommended approach because of the following points:
* Your classes will be cluttered with test code in main method. All those test codes will be packaged into the final product.
* If you have a lots of classes to test, you need to run the main() method of every class. This requires some extra coding effort.
* If you want the test results to be displayed in a GUI, you will have to write code for that GUI.
* If you want to log the results of tests in HTML format or text format, you will have to write additional code.
* If one method call fails, next method calls won?t be executed. You will have to work-around this.
* If you start working on a project created by some other team in your organization, you may see an entirely different approach for testing. That will increase your learning time and things won?t be standard.

Read More Answers.

Question # 18

Why Not Just Use System.out.println() for Unit Testing?

Answer:-

Inserting debug statements into code is a low-tech method for debugging it. It usually requires that output be scanned manually every time the program is run to ensure that the code is doing what's expected.
It generally takes less time in the long run to codify expectations in the form of an automated JUnit test that retains its value over time. If it's difficult to write a test to assert expectations, the tests may be telling you that shorter and more cohesive methods would improve your design.

Read More Answers.

Question # 19

Under What Conditions Should You Test set() and get() Methods?

Answer:-

This is a good question for a job interview. It shows your experience with test design and data types.
Tests should be designed to target areas that might break. set() and get() methods on simple data types are unlikely to break. So no need to test them.
set() and get() methods on complex data types are likely to break. So you should test them.

Read More Answers.

Question # 20

Do You Need to Write a Test Class for Every Class That Need to Be Tested?

Answer:-

This is a simple question. But the answer shows your organization skills.
The technical answer is no. There is no need to write one test class for each every class that need to be tested. One test class can contain many tests for many test target classes.
But the practical answer is yes. You should design one test class per test target class for low level basic tests. This makes your test classes much easier to manage and maintain.
You should write separate test classes for high level tests that requires multiple target classes working together.

Read More Answers.

Global Guideline . COM

Question # 21

What Is JUnit TestCase?

Answer:-

JUnit TestCase is the base class, junit.framework.TestCase, used in JUnit 3.8 that allows you to create a test case. TestCase class is no longer supported in JUnit 4.4.
A test case defines the fixture to run multiple tests. To define a test case
* Implement a subclass of TestCase
* Define instance variables that store the state of the fixture
* Initialize the fixture state by overriding setUp
* Clean-up after a test by overriding tearDown

Each test runs in its own fixture so there can be no side effects among test runs. Here is an example:

```
import junit.framework.*;
public class MathTest extends TestCase {
   protected double fValue1;
   protected double fValue2;
   protected void setUp() {
      fValue1= 2.0;
      fValue2= 3.0;
   }

   public void testAdd() {
      double result= fValue1 + fValue2;
      assertTrue(result == 5.0);
   }
}
```

Read More Answers.

Question # 22

How to Run Your JUnit 4.4 Tests with a JUnit 3.8 Runner?

Answer:-

I am not sure why you have to do this. But if you want to, you can use the junit.framework.JUnit4TestAdapter class included in JUnit 4.4 JAR file. Here is sample code:

```
import junit.framework.Test;
import junit.textui.TestRunner;
import junit.framework.JUnit4TestAdapter;
public class JUnit3Adapter {
  public static void main (String[] args) {
    Test adaptedTest = new JUnit4TestAdapter(HelloTest.class);
    TestRunner.run(adaptedTest);
  }
}
```

Classes junit.framework.Test, junit.textui.TestRunner and junit.framework.JUnit4TestAdapter are included in the JUnit 4.4 JAR file. You don't need to include the JUnit 3.8 JAR file in your CLASSPATH.

Read More Answers.

Question # 23

What Are JUnit 3.8 Naming Conventions?

Answer:-

JUnit 3.8 suggests the following naming conventions:
Test Case Class: Named as [classname]Test.java, where "classname" is the name of the class that is being tested. A test case class define the fixture to run multiple tests. A test case class must be subclass of junit.framework.TestCase.
Test Method: Named test[XXX], where "XXX" is any unique name for this test. A test method name should be prefixed with "test" to allow the TestSuite class to extract it automatically. A test method must be declared as "public".
Test Suite: Can be named any way you want to. But Eclipse uses AllTests.java as the name. A test suite is a collection of test cases.

Read More Answers.

Question # 24

How Many Test Runners Are Supported in JUnit 3.8?

Answer:-

JUnit 3.8 supports 3 test runners:
junit.textui.TestRunner - A command line based tool to run tests. TestRunner expects the name of a TestCase class as argument. If this class defines a static suite method it will be invoked and the returned test is run. Otherwise all the methods starting with "test" having no arguments are run.
junit.awtgui.TestRunner - An AWT based user interface to run tests. Enter the name of a class which either provides a static suite method or is a subclass of TestCase. TestRunner takes as an optional argument the name of the testcase class to be run.
junit.swingui.TestRunner - A Swing based user interface to run tests. Enter the name of a class which either provides a static suite method or is a subclass of TestCase. TestRunner takes as an optional argument the name of the testcase class to be run.
All 3 runners can be executed directly by JVM with a class name as an argument

Read More Answers.

Question # 25

What Is a JUnit Test Fixture?

Answer:-

A test 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

Copyright © https://globalguideline.com

environment in which tests are run so that results are repeatable. Examples of fixtures:
* Loading a database with a specific, known set of data
* Copying a specific known set of files
* Preparation of input data and setup/creation of fake or mock objects

In other word, creating a test fixture is to create a set of objects initialized to certain states.
If a group of tests requires diferent test fixtures, you can write code inside the test method to create its own test fixture.
If a group of tests shares the same fixtures, you should write a separate setup code to create the common test fixture.

Read More Answers.

Question # 26

Can You Explain the Life Cycle of a JUnit 3.8 Test Case Class?

Answer:-

A JUnit 3.8 test case class contains a setUp() method, a tearDown() method and multiple testXXX() methods. When calling a test runner to run this test class, the runner will execute those methods in a specific order giving the test case class an execution life cycle like this:
setUp()
testXXX1()
tearDown()
setUp()
testXXX2()
tearDown()
setUp()
testXXX3()
tearDown()

Read More Answers.

Question # 27

How To Write Setup Code to Run Once for All Tests in a Test Class?

Answer:-

From the previous question, you know that if you write a setup code under the "@Before" annotation, it will be executed many times: once per each test in the test class.
Is there any way to write a setup code that will be executed only once for all tests in a single test class? The answer is YES. Here is how the JUnit FAQ answers this question in details:
The desire to do this is usually a symptom of excessive coupling in your design. If two or more tests must share the same test fixture state, then the tests may be trying to tell you that the classes under test have some undesirable dependencies.
Refactoring the design to further decouple the classes under test and eliminate code duplication is usually a better investment than setting up a shared test fixture.
But if you must...
You can add a @BeforeClass annotation to a method to be run before all the tests in a class, and a @AfterClass annotation to a method to be run after all the tests in a class.

Read More Answers.

Question # 28

Can You Explain the Life Cycle of a JUnit 4.4 Test Class?

Answer:-

A JUnit 4.4 test class contains a @Before method, an @After method and multiple @test methods. When calling a test runner to run this test class, the runner will execute those methods in a specific order giving the test class an execution life cycle like this:
@Before
@Test XXX1
@After
@Before
@Test XXX2
@After
@Before
@Test XXX3
@After

Read More Answers.

Question # 29

How To Group Test Cases Class using JUnit TestSuite?

Answer:-

Usually, there are many classes to be tested in a Java application. For each Java class in the application you need to write a JUnit test case class comtaining multiple test methods.
You could call a JUnit runner to run each JUnit test case class manually. But you should group all JUnit test case clases into a test suite with JUnit TestSuite. The following code and notes are provided by Varun Chopra and valid for JUnit 3.8 with some corrections.
To group all test case classes together and run them a single unit, you should create a new class named like AllTests.java. In this class you must create a "public static Test suite()" method, which returns a TestSuite object as a container of test case classes.

Read More Answers.

Question # 30

How Do You Test a private Method?

Answer:-

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

Global Guideline. COM

class.
To resolve this problem, you have to perform unit testing manually. Or you have to change your method from "private" to "protected".

Read More Answers.

Question # 31

How Do You Test a protected Method?

Answer:-

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

Read More Answers.

Question # 32

Can You Write a Simple Class for JUnit Testing in 1 Minute?

Answer:-

You need to write a class with less than 10 lines. Here is a nice example provided in the article "Writing a JUnit Test in 5 minutes" by Kamal Mettananda with some minor changes:

```
package com.parcelhouse.myproj;
public class Calc {
    public int add(int a, int b) {
        // errorneous method
        return a+b+1;
    }
    public int multiply(int a, int b) {
        return a*b;
    }
}
```

Read More Answers.

Question # 33

How Do You Test Classes That Must Be Run in a J2EE Container?

Answer:-

To test classes that must be run in a J2EE container (e.g. servlets, EJBs), you should refactor 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 to test J2EE components in their natural environment.

Read More Answers.

Question # 34

How to creating a Test Suite using JUnit in Eclipse?

Answer:-

There are four ways to create a JUnit test suite class in Eclipse with JUnit plugin: org.junit_3.8.1. First, select the directory (usually unittests/) that you wish to create the test suite class in.
1. Select File > New > Other... > Java > JUnit > JUnit Test Suite.
2. Select the arrow of the button in the upper left of the toolbar. Select Other... > Java > JUnit > JUnit Test Suite,
3. Right click on a package in the Package Explorer view in the Java Perspective, and select Other... > Java > JUnit > JUnit Test Suite, or
4. You can create a normal Java class as shown in the Eclipse tutorial, but include junit.framework.TestSuite as the super class of the test class you are creating.

Read More Answers.

Question # 35

Can You Describe Steps of Creating Test Case Classes in Eclipse?

Answer:-

There are detailed steps to create a test case class in Eclipse with JUnit plugin: org.junit_3.8.1.
1. Use the Browse button to search for a different super class. The default super class is junit.framework.TestCase.
2. Check which method stubs you would like to create. You can create a main method, setUp(), tearDown(), or a constructor(), but all of these are optional. A constructor is only run when the test case class is first instantiated, but the setUp() and tearDown() methods are run before and after, respectively, each test case is run.
3. You can browse the application that you are creating for a class that you wish to test, or this could be left blank if you will generate the class while creating while creating the test.
- If you selected a "Class Under Test" you can click the Next button, otherwise click Finish. You will be able to select which methods in the class under test that you want to write test cases for. The method signatures will be created for you. Click Finish. The new test case class will be open in the editor.
- This test class demonstrates the basic functionality of the setUp() and tearDown() methods, and gives example test cases. The testForException() method demonstrates how to test that an exception is properly thrown.
Note: All source methods in the class under test must be public or protected, not private, in order to be tested by JUnit. If the method in the class under test is private, the test class must be in the same package.

Read More Answers.

Question # 36

How to Run a JUnit Test Case in Eclipse?

Answer:-

There are three ways to run JUnit Test Cases or Test Suites in Eclipse with JUnit plugin: org.junit_3.8.1.

Global Guideline . COM

1. You can right click on the test case class or test suite class and select Run As > JUnit Test.
2. You can select a test case or suite and click the arrow on the icon or select Run from the toolbar, and select Run As > JUnit Test.
3. You can select a test case or suite and click the arrow on the icon or select Run from the toolbar, and select Run... From here you will create a new JUnit test configuration, and name it. You can choose to run a single test case, or run all test cases in a project or folder.

Read More Answers.

Question # 37

How To Create Test Class in Eclipse?

Answer:-

There are five ways to create a JUnit test case class in Eclipse with JUnit plugin: org.junit_3.8.1. First, select the directory (usually unittests/) that you wish to create the test case class in.
1. Select File > New > JUnit Test Case.
2. Select the arrow of the button in the upper left of the toolbar. Select JUnit Test Case.
3. Right click on a package in the Package Explorer view in the Java Perspective, and select JUnitTestCase.
4. Click on the arrow of the icon in the toolbar. Select JUnit Test Case.
5. You can create a normal Java class as shown in the Eclipse tutorial, but include junit.framework.TestCase as the super class of the test class you are creating.

Read More Answers.

Question # 38

How simple is too simple to break?

Answer:-

The general philosophy is this: if it can't break on its own, it's too simple to break.
First example is the getX() method. Suppose the getX() method only answers the value of an instance variable. In that case, getX() cannot break unless either the compiler or the interpreter is also broken. For that reason, don't test getX(); there is no benefit. The same is true of the setX() method, although if your setX() method does any parameter validation or has any side effects, you likely need to test it.

Read More Answers.

Question # 39

What Do You Do When a Defect Is Reported?

Answer:-

Test-driven development generally lowers the defect density of software. But we're all fallible, so sometimes a defect will slip through. When this happens, write a failing test that exposes the defect. When the test passes, you know the defect is fixed!
Don't forget to use this as a learning opportunity. Perhaps the defect could have been prevented by being more aggressive about testing everything that could reasonably break.
Or perhaps there are other places in the application that have the similar code that might break too.

Read More Answers.

Question # 40

How Often Should You Run Your JUnit Tests?

Answer:-

Run all your unit tests as often as possible, ideally every time the code is changed. Make sure all your unit tests always run at 100%. Frequent testing gives you confidence that your changes didn't break anything and generally lowers the stress of programming in the dark.
For larger systems, you may just run specific test suites that are relevant to the code you're working on.
Run all your acceptance, integration, stress, and unit tests at least once per day (or night).
If you're using Eclipse, be sure to check out David Saff's continuous testing plug-in.

Read More Answers.

Question # 41

Do You Have To Write a Test for Everything?

Answer:-

No, just test everything that could reasonably break.
Be practical and maximize your testing investment. Remember that investments in testing are equal investments in design. If defects aren't being reported and your design responds well to change, then you're probably testing enough. If you're spending a lot of time fixing defects and your design is difficult to grow, you should write more tests.
If something is difficult to test, it's usually an opportunity for a design improvement. Look to improve the design so that it's easier to test, and by doing so a better design will usually emerge.

Read More Answers.

Question # 42

When Objects Are Garbage Collected After a Test Is Executed?

Answer:-

My guess would be that all objects used in a test will be ready for Java garbage collector to release them immediately after the test has been executed.
By design, the tree of Test instances is built in one pass, then the tests are executed in a second pass. 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.
Therefore, if you allocate external or limited resources in a test, you are responsible for freeing those resources. 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.
If this is true, the JUnit runner should be improved to stop building all test instances before executing any tests. Instead, the JUnit runner should just take one test at a time, build an instance of this test, execute the test, and release the test when the execution is done.

Global Guideline . COM

Read More Answers.

Question # 43

How Do You Test a Method That Does not Return Anything?

Answer:-

You need to follow the logic below to answer this question:
* If a method is not returning anything through the "return" statement (void method), it may return data through its arguments. In this case, you can test the data returned in any argument.
* Else if a method is not returning any data through its arguments, it may change values of its instance variables. In this case, you can test changes of any instance variables.
* Else if a method is not changing any instance variable, it may change values of its class variables. In this case, you can test changes of any class variables.
* Else if a method is not changing any class variable, it may change external resources. In this case, you can test changes of any external resources.
* Else if a method is not changing any external resources, it may just doing nothing but holding the thread in a waiting status. In this case, you can test this waiting condition.
* Else if a method is not holding the thread in waiting status, then this method is really doing nothing. In this case, there is no need to test this method. :-)

Read More Answers.

Question # 44

When Should Unit Tests Should Be Written In Development Cycle?

Answer:-

If you are a TDD (Test-Driven Development) believer, you should write unit test before writing the code. Test-first programming is practiced by only writing new code when an automated test is failing.
Good tests tell you how to best design the system for its intended use. They effectively communicate in an executable format how to use the software. They also prevent tendencies to over-build the system based on speculation. When all the tests pass, you know you're done!
Whenever a customer test fails or a bug is reported, first write the necessary unit test(s) to expose the bug(s), then fix them. This makes it almost impossible for that particular bug to resurface later.
Test-driven development is a lot more fun than writing tests after the code seems to be working. Give it a try!

Read More Answers.

Question # 45

How Do You Launch a Debugger When a Test Fails?

Answer:-

Configure the debugger to catch the java.lang.AssertionError exception and suspend the execution.
Call the TestRunner to run the test under the debugger.
When the test fails again, The debugger will suspend the execution and start the debug mode.
Now you are ready to use debugger commands to find the code issue that causing the fail.
How you configure this depends on the debugger you prefer to use.
Most Java debuggers provide support to stop the program when a specific exception is raised.

Read More Answers.

Global Guideline . COM

Testing Most Popular Interview Topics.

1 : Manual Testing Frequently Asked Interview Questions and Answers Guide.

2 : QTP Frequently Asked Interview Questions and Answers Guide.

3 : Software QA Frequently Asked Interview Questions and Answers Guide.

4 : QA Testing Frequently Asked Interview Questions and Answers Guide.

5 : Mobile Testing Frequently Asked Interview Questions and Answers Guide.

6 : Database Testing Frequently Asked Interview Questions and Answers Guide.

7 : Test Cases Frequently Asked Interview Questions and Answers Guide.

8 : Localization Testing Frequently Asked Interview Questions and Answers Guide.

9 : Software Testing Frequently Asked Interview Questions and Answers Guide.

10 : WinRunner Frequently Asked Interview Questions and Answers Guide.

About Global Guideline.

Global Guideline is a platform to develop your own skills with thousands of job interview questions and web tutorials for fresher's and experienced candidates. These interview questions and web tutorials will help you strengthen your technical skills, prepare for the interviews and quickly revise the concepts. Global Guideline invite you to unlock your potentials with thousands of **Interview Questions with Answers** and much more. Learn the most common technologies at Global Guideline. We will help you to explore the resources of the World Wide Web and develop your own skills from the basics to the advanced. Here you will learn anything quite easily and you will really enjoy while learning. Global Guideline will help you to become a professional and Expert, well prepared for the future.

* This PDF was generated from https://GlobalGuideline.com at **November 29th, 2023**

* If any answer or question is incorrect or inappropriate or you have correct answer or you found any problem in this document then don't hesitate feel free and e-mail us we will fix it.

You can follow us on FaceBook for latest Jobs, Updates and other interviews material.
www.facebook.com/InterviewQuestionsAnswers

Follow us on Twitter for latest Jobs and interview preparation guides
https://twitter.com/InterviewGuide

Best Of Luck.

Global Guideline Team
https://GlobalGuideline.com
Info@globalguideline.com