Unit Testing & JUnit

- **What is unit testing?**

UNIT TESTING is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

- **What is the difference between manual testing and automated testing?**

- Manual Testing is done manually by QA analyst (Human) whereas Automation Testing is done with the use of script, code and automation tools (computer) by a tester.
- Manual Testing process is not accurate because of the possibilities of human errors whereas the Automation process is reliable because it is code and script based.
- Manual Testing is a time-consuming process whereas Automation Testing is very fast.
- Manual Testing is possible without programming knowledge whereas Automation Testing is not possible without programming knowledge.
- Manual Testing allows random Testing whereas Automation Testing doesn't allow random Testing.

- **Is it necessary to write the test case for every logic? If yes, why**

No, we should write the test case only for that logic that can be reasonably broken.

- **What are the features of JUnit?**

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 are the important JUnit annotations? And its usage in coding**

The test runner is used to execute the test cases.

- @Test
- @BeforeClass
- @Before
- @After
- @AfterClass

➢ **What does Assert class?**

Assert class provides methods to test the test cases.

➢ **What is Code Coverage?**

Code coverage is a software testing metric that determines the number of lines of code that is successfully validated under a test procedure, which in turn, helps in analyzing how comprehensively a software is verified.

Developing enterprise-grade software products is the ultimate goal of any software company. However, to accomplish this goal, companies have to ensure that the software they develop meets all the essential quality characteristics – **correctness, reliability, effectiveness, security, and maintainability.** This can only be possible by thoroughly **reviewing the software product**.

➢ **What are the best practices to perform Unit Testing?**

A test process is good if:

Unit Testing & JUnit

It is automated (on CI)

Tests are only useful if they are executed in a timely manner. The best option is to use continuous integration which will constantly run your tests, for example, on each commit. Otherwise, it is easy to forget to run tests, which makes them useless.

Tests are written during development, not after

TDD (when you write tests before writing code) is great, but from the beginning, it might not be that easy to foresee what your module should look like, what the structure of the classes will be, and so on.

So, if one can't write tests first — that's OK. What is important is to set up tests as early in the development as possible and to not delay them until the end.

The reason is that tests help you to write clean code. Separate concerns, use interfaces to hide implementation details or some platform-specifics. If you delay writing tests, you can find yourself in a position where some code is not testable and it would be very tempting to hack it around.

Tests are added for each defect/case found

When write tests, you don't need to take care of all the situations that might theoretically happen (we'll talk about it in detail below).

The most important thing is to reflect business use cases and add tests continuously for any other requirement or defect found. Especially for defects. Because this way, you can verify before fixing that there is a case failing and check that after the fix, the test actually passed.

- **What is Mocking?**

Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behavior of the object you want to replace the other objects by mocks that simulate the behavior of the real objects. This is useful if the real objects are impractical to incorporate into the unit test

Unit Testing & JUnit