A unit test is a type of software test that focuses on components of a software product.

The purpose is to ensure that each unit of software code works as expected. A unit can be a function, method, module, object, or other entity in an application’s source code.

The objective (цель) of a unit test is to test an entity in the code, ensure that it is coded correctly with no errors, and that it returns the expected outputs for all relevant (уместный) inputs.

Unit tests are typically created by developers during the coding phase of a project, and are written as code that exists in the codebase alongside the application code it is testing.

The Unit Testing Techniques are mainly categorized into three parts which are:

-**Black box** testing that involves testing of user interface along with input and output. Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability (надежность) issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

Advantages: Testers do not require technical knowledge, programming or IT skills, Testers do not need to learn implementation details of the system, Tests can be executed by crowdsourced or outsourced testers, Low chance of false positives (Низкая вероятность ложных срабатываний), Tests have lower complexity, since they simply model common user behavior.

Disadvantages: Difficult to automate, Requires prioritization, typically infeasible to test all user paths(Требуется расстановка приоритетов, обычно невозможно протестировать все пути пользователей), Difficult to calculate test coverage(Трудно рассчитать тестовое покрытие), If a test fails, it can be difficult to understand the root cause of the issue, Tests may be conducted at low scale or on a non-production-like environment(Испытания могут проводиться в небольших масштабах или в непроизводственной среде)

Black box testing can be applied to three main types of tests: functional, non-functional, and regression testing.

-**White box** testing that involves testing the functional behavior of the software application,White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems. White box testing is an essential part of automated build processes in a modern Continuous Integration/Continuous Delivery (CI/CD) development pipeline.White box testing provides inputs and examines outputs, considering the inner workings of the code. White box testing involves complete knowledge of the inner workings of a system under test

Advantages: Ability to achieve complete code coverage, Easy to automate, Reduces communication overhead (накладные расходы/непонимание) between testers and developers

Disadvantages: Sensitive to changes in code base, automation requires expensive maintenance. Cannot test expected functionality that does not exist in the codebase, Cannot test from the user’s perspective.

-**Gray box** testing that is used to execute test suites, test methods, test cases and performing risk analysis.Gray box testing is a method you can use to debug software and evaluate vulnerabilities(уязвимости). In this method, the tester has limited knowledge of the workings of the component being tested. This is in contrast to black box testing, where the tester has no internal knowledge, and white box testing, where the tester has full internal knowledge.

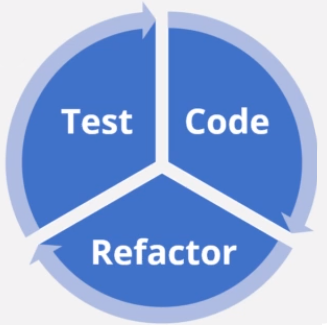
**Many unit testing frameworks** exist that help developers manage and execute unit tests. Test-driven development (TDD) is a common approach to unit testing.

**TDD**

Test driven development - it is primarily used for unit testing by the development team. It is not a methodology like waterfall or Agile that talks overall (общий) software development cycle, but **it's more of programming style or practice** that talks about how to write the code in iterations (iterations - means during software development iteration). There are mainly three steps in TDD: **Test, Code, Refuctor**.

1)- The developer would first write an unit test that would describe the functionality to be developed. The test will execute and it will fail as expected.

2) - The developer would then write just enough code to make the test pass.

3) - The developer than would refactor the code, to remove any code duplication or any inconsistency (непоследовательности) in the code. 

Then the same process is repeated.

This way the developer keeps accumulating unit tests that form the safety net (подстраховку) for the code quality. This practice helps developers to identify the defects (bags) in the developers phase itself. That can reduce the amount of defects that would escape to the later phases of software development, where it can become very costly to fix them. Overall software quality increases because now any change in the code may fail the unit test, and the defects will be caught during development (code phase) itself. In the end, this helps reduce the overall cost at the expense of a slightly delayed development cycle.

Effective unit testing typically:

-Runs each test case in an isolated manner, with “stubs” or “mocks” used to simulate external dependencies. This ensures the unit tests only considers the functionality of the current unit under test.

-Does not test every line of code, focusing on critical features of the unit under test. In general, unit testing should focus on code that affects the behavior of the overall software product.

-Verifies each test case using criteria determined in code, known as “assertions”. The testing framework uses these to run the test and report failed tests.

-Runs frequently and early in the development lifecycle.