A Short Introduction to Junit

Junit = a test-framework for unit tests in Java.

Unit = a small piece of code, usually a function or method.

Test Philosophy:

Tests are organised according to the corresponding classes.

Example: if you have a “StudentClass”, make sure you have a corresponding test class “StudentClassTest”.

-> see my example.

Assert Class in JUnit

The „Assert“ class is one of the main classes in JUnit which asserts (“verifies”) your code. It offers a number of methods you can use: assertTrue, assertEquals, assertNotNull, etc. Check out the JUnit API.

How do I identify a test case?

- ➔ Remember: Unit tests are on a micro level, they map to singular methods, never to a use case.
- ➔ Identify important methods in your class. These are probably unit test candidates.
- ➔ Setter/Getter methods are usually not unit test cases, unless they have a central role in your class.

Test Coverage

We use the term “test coverage” to talk about the percentage of test cases that cover your application. A rule of thumb says, the coverage should be around 90% to have a stable / test proofed system.

Organising Test Cases

You can bundle unit tests to a test suite (a row of tests that cover a function of the application). You can implement a test run class, which will run all your test classes in a defined order.

White vs Black Box Testing

White box testing = user knows the code (or program)

Black box testing = user doesn't know the code

Test-Driven Development (TDD)

This is a development philosophy, a way how to develop applications.

Main idea: you start with your test case first, then you write the code.

We didn't follow this approach in our module project. But maybe you have seen this approach at your job?