Testing

mocks
stubs
expected
Junit
@Test
hamcrest
integration-testing
positive-tests
test-driven-development
unit-testing
negative-tests
@Before

Test Driven Development

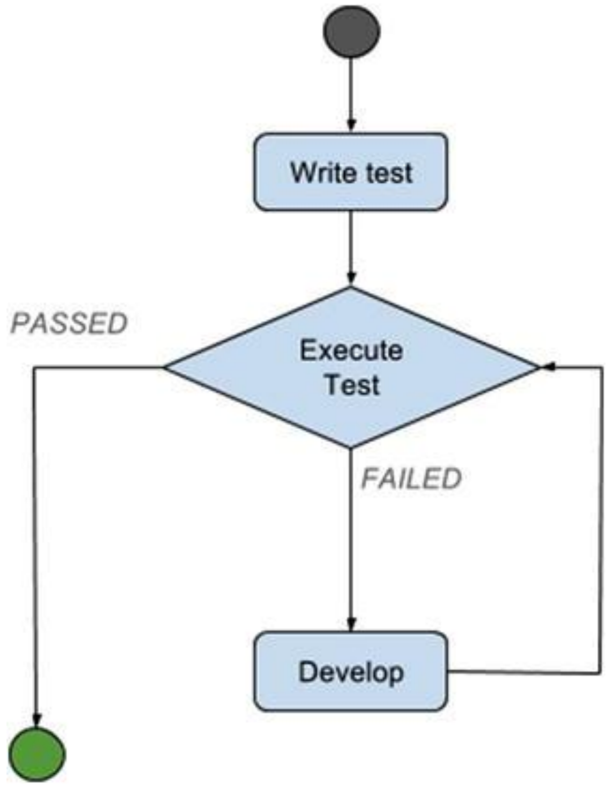
- Test-Driven Development (TDD) is writing tests before writing any functional code, and then writing only the least possible amount of code required to make the tests pass

- Quality cannot be tested in

- TDD philosophy, tests drive the design of the software by thinking how it could be tested

- KISS is an acronym for "Keep it simple, stupid" as a design principle noted by the U.S. Navy in 1960. The KISS principle states that most systems work best if they are kept simple rather than made complicated;

Test Driven Development



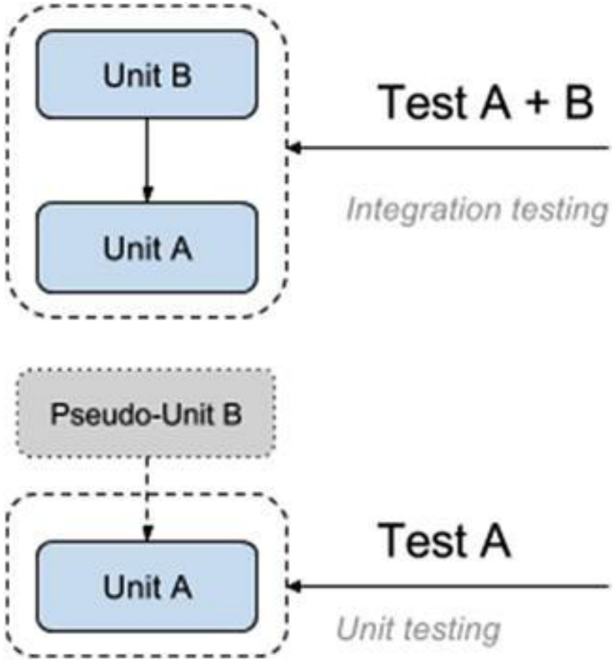
Unit and Integration Testing

- Unit testing implies testing the smallest testable parts of an application individually and independently, isolated from any other units that might affect their behavior in an unpredictable way.

- The tests are written by developers, and the recommended practice is to cover every method in a class with positive and negative tests. Positive tests are the ones that test valid inputs for the unit, while negative tests test invalid inputs for the unit.

- Running a suite of unit tests together in a context with all their real dependencies provided is called integration testing

Unit and Integration Testing



Pseudo-dependencies

- The pseudo-dependencies can be stubs or mocks. Both perform the same function, to replace a real dependency, but they way they are created is what sets them apart

- Stubs are created by the developer; they do not require extra dependencies. A stub is a concrete class implementing the same interface as the original dependency of the unit being tested. They should be designed to exhibit a small part or the whole behavior of the actual dependency.

- The mock object will implement the dependent interface on the fly. Before a mock object is generated, the developer can configure its behavior: what methods will be called and what will they return.

Stubs

- Advantages
  - enables unit-testing
  - requires no further dependencies
- Disadvantages
  - if the interface changes, the stub implementation has to change
  - all methods have to be implemented

mocks
stubs
expected
Junit
@Test
hamcrest
integration-testing
positive-tests
test-driven-development
unit-testing
negative-tests
@Before