Unit Testing & Mocking

Agenda

1 OVERVIEW / INTRODUCTION

2 UNIT TESTING

3 MOCKING

OVERVIEW / INTRODUCTION

What is software testing

PROCESS WITH INTENT OF FINDING SOFTWARE BUGS

- Part of the development lifecycle
- It can be static or dynamic
- Planning and preparation must precede the actual testing
- Evaluation and check the result

TOOL TO VERIFY THAT THE PROGRAM WORKS AS EXPECTED

- Meets the business requirements
- The software works as expected
- Can be implemented with the same characteristic

Cost of NOT Testing

COMPANY CAN LOSE

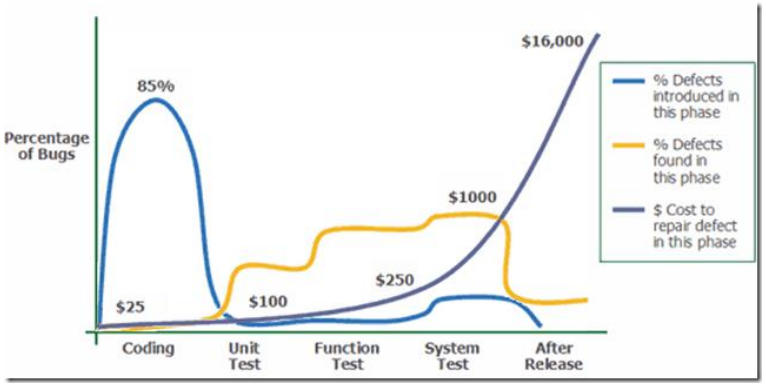
- Reputation
- Human life
- Pollution
- Money

HOWEVER...

- Different Systems
- Different level of risks
- Different effects

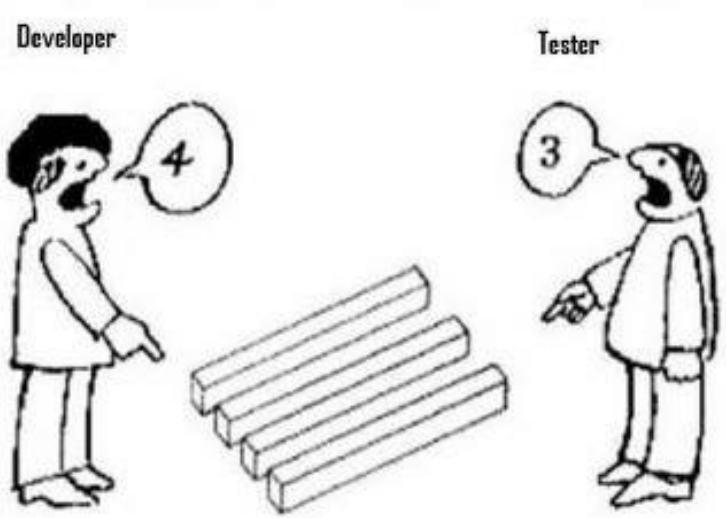
Early Testing

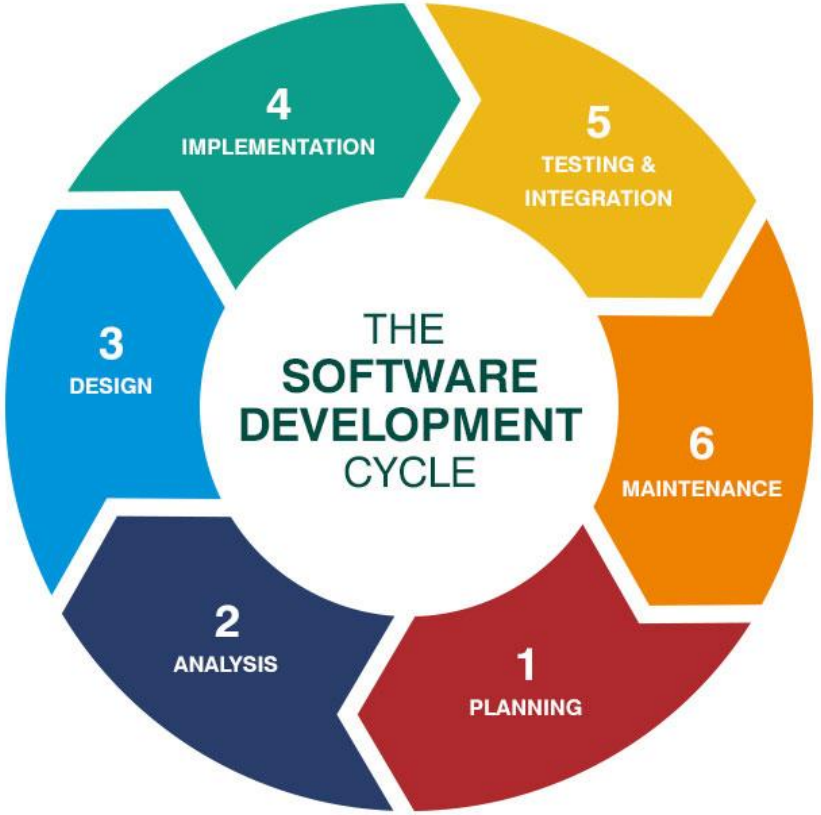
Phase	Relative Cost
Requirements	1
Implementation	10
Testing	100
System Test	1 000
Live System	10 000



The software development cycle

- When should we test? – All the time
- Who need to test? – Everybody
- Who is responsible for Quality? – The whole team

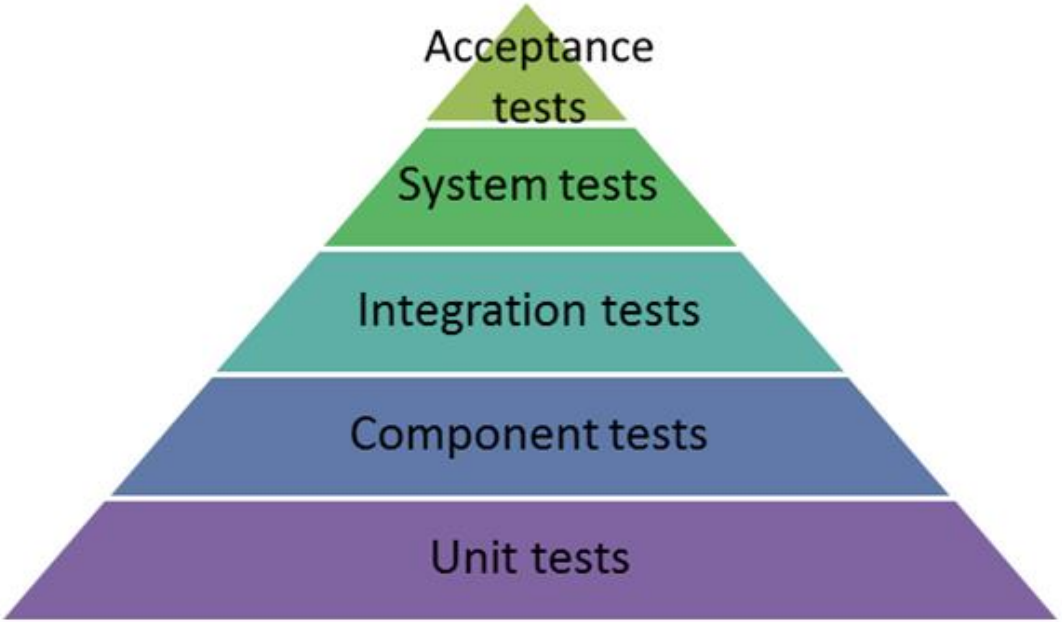




UNIT TEST

Testing pyramid

IT IS NOT ENOUGH TO HAVE ONLY UNIT TESTS



Unit testing – Why? What? How?

MAKES DEVELOPMENT EASIER

- Developers can become more confident
- Immediate feedback about code changes

LOWER MAINTENANCE COST

- Saves effort when one needs to identify the root cause of broken code
- Documents use cases at low level
- Points out bugs much earlier than they could cause bigger issues

CLASSIC DEFINITION OF UNIT TEST

- A unit test is a piece of code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward.
- If the assumptions turn out to be wrong, the unit test has failed.
- A “unit” is a method or function.

CHARACTERISTICS OF A GOOD UNIT TEST

- **F**ast
- **I**ndependent
- **R**epeatable
- **S**elf-validating
- **T**imely/Thorough

Unit testing frameworks

EXPECTATIONS AGAINST FRAMEWORKS

- Help in writing test easily and in a structured manner
- Provide a way to execute one or all of the unit tests
- Present the result of the test runs somehow

UNIT TESTING FRAMEWORKS FOR JAVA

- JUnit 5
- TestNG

JUNIT 4 BASIC ANNOTATIONS

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

Basic Rules to follow when writing Unit Tests

NAMING CONVENTIONS

- Name the test class like `[SystemUnderTest]Test`
- Name the test methods like `test[TestedMethod]Should[DoSomething]When[Condition]()`
- Name the tested object's variable conventional, like `underTest`

TEST METHOD STRUCTURE

- GIVEN: initialize a state the tested method should run in
- WHEN: call the tested method
- THEN: verify the new state

HOW MANY TEST METHODS SHOULD YOU WRITE?

- At least 1 test method for all method that contains any kind of logic
- However the good approach is to have as many test method as the cyclomatic complexity of the tested method

FURTHER RULES

- Never initialize a state in the Before method that are not needed for ALL your test methods!
- Keep it simple
- Keep it easily understandable
- Keep it conventional
- The unit test should also describe the logic the production class implements: helps in understanding the code

MOCKING

Unit Testing – Principles

UNIT TEST RULES BY MICHAEL FEATHERS

- A test is not a unit test if
  - It talks to the database
  - It communicates across the network
  - It touches the file system
  - It can’t run correctly at the same time as any of your other unit tests
  - You have to do special things to your environment (such as editing config files) to run it

UNIT TESTS ARE NOT INTEGRATION TESTS

UNIT TEST

- Only one unit in scope
- Test runs quickly
- Specific errors
- Only the unit must be initialized
- Change in a unit affects only one test

INTEGRATION TEST

- Multiple participants in an interaction
- Test can run longer
- Hard to localize cause of failure
- Additional configuration and setup
- Changes in dependencies affect more than one test

How to deal with dependencies?

STUBS / MOCKS

- Replacement for an existing dependency in the system
- The Unit Test can have control over
- The SUT interacts with stubs instead of real dependencies
- Stubs
    - Stubs are silent contributors in testing
    - We cannot record interactions between SUT and Stubs
- Mocks
    - Also replaces a real object
    - Allows verifying the calls (interactions)
    - Implements the same interface as the replaced object
    - Can be controlled, created and injected into the system under test by the unit test

MOCKING FRAMEWORKS

- A set of programmable APIs
- Allow creating Mock and Stub Objects in an easy way
- Prevents creating Mocks and Stubs manually

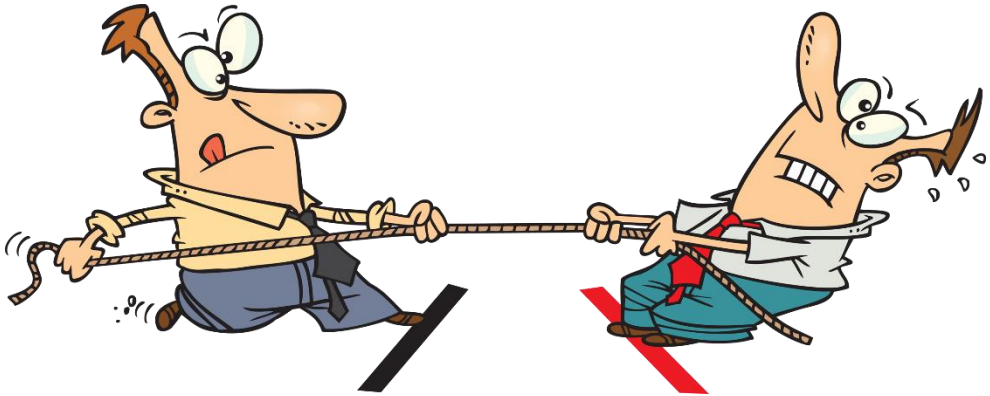
Two groups of Unit Test writers

CLASSICITS

- After the exercise phase they check the collected results and the state of the tested object
- Use assertions

MOCKISTS

- Before the exercise phase they expects some specific behavior and after the exercise phase they verify if it happened
- Use mock objects



Mockito

USEFUL METHODS

- org.mockito.Mockito.*mock*
- org.mockito.Mockito.*when*
- org.mockito.Mockito.*verify*
- org.mockito.Mockito.*times*
- org.mockito.Mockito.*verifyNoMoreInteractions*

Unit Testing's effects on production code

WHAT MAKES CODE EASILY TESTABLE?

- Clean dependency hierarchy
- Clean methods (simple, not too complex ones)
- Keeping Test Unfriendly Features on low degree

TEST UNFRIENDLY FEATURES

- Access to database, filesystem, network
- Side effecting APIs (like GUIs)
- Lengthy computations
- Static variable usage

TEST UNFRIENDLY CONSTRUCTS

- Final methods, classes
- Static, private methods
- Static initialization expression or blocks
- Constructors
- Object initialization blocks
- New expressions

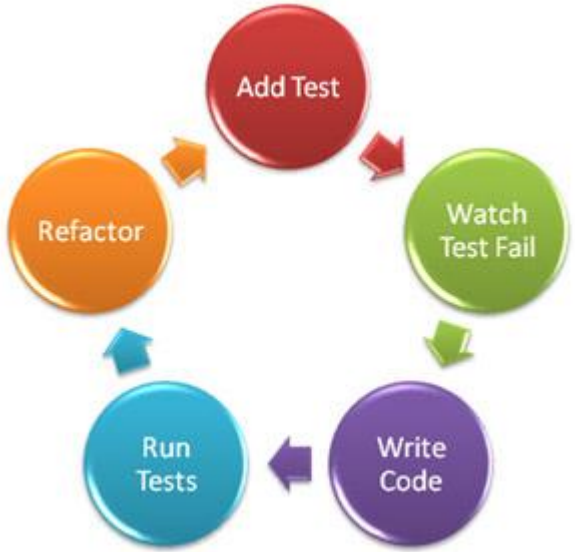
DESIGNING TO TESTABILITY IMPROVES CODE AND DESIGN QUALITY

- Cleans dependencies
- Decreases complexity
- Highlights responsibilities

Test Driven Development

TEST FIRST, THEN IMPLEMENT PRODUCTION CODE

- TDD one step
    - Create production class/method
    - Create unit test class/method
    - Test a requirement
    - Run test, see if it fails
    - Satisfy requirement
    - Runt test, see if it passes
    - Refactor
- Someones love it, others hate it
- Give it a try and use if you like, not otherwise



THANK YOU FOR YOUR ATTENTION