Testing 101

Alex Poovathingal

Gozefo.com

January 11, 2018

Introduction

- Coding standards
- Testing
- Junit
- Mockito
- Dependency Injection

Disclaimer - This talk contains unsolicited advice on programming standards* and testing. I do not claim to have followed these and shall not be held liable for any issuesof whatever nature which may arise as a result of following these advices.
* Not Really!!

Useless Documentation

```
/**
* Created by alex on 25/3/16.
*/
public class SomeNounDao.java {
}

/**
* Do something after this other thing
*
* @param randomObject Random Object
* @param userId     User Id
* @return Boolean
*/
public Boolean doSomethingAfterThisOtherThing(
        RandomObject randomObject,
        String userId) {
}
```

Better Programming Constructs

- Loops considered harmful
- Maps, ForEach, Times, Filters For a Better World
- No more arrayLists or arrays

A world without Nulls

Before

```
SomeObject k = a.getSomething();
// Code continues for ever without another usage of a
```

After

```
SomeObject k;
if (a != null) {
    k = a.getSomething();
}
// Code continues for ever without another usage of a
```

Over engineering

- Not everything needs to scale (Most things doesn't need to scale).
- Readability vs Scalability

Tech Debt

- Tech Debt is real(Based on true story)
- Less lines of code = Less number of lines which could be buggy
- Style checks
- If without an Else

Functions Functions Functions

- There is a thing called function
- Number of exit points
- **Pure functions**
- Functions returns null

What is testing

- Identify the quality of the software
- Find bugs and missing functionalities

Types of Testing

Types of Testing

- Manual Testing

Types of Testing

- Manual Testing
- Unit Testing

Types of Testing

- Manual Testing
- Unit Testing
- Integration Testing

Types of Testing

- Manual Testing
- Unit Testing
- Integration Testing
- Functional Testing

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation
- Testable code is well designed code

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation
- Testable code is well designed code
- Forces you to slow down and think.

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation
- Testable code is well designed code
- Forces you to slow down and think.
- Defends against other programmers

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation
- Testable code is well designed code
- Forces you to slow down and think.
- Defends against other programmers
- Fearless programming. Refactor!!

Why write tests

- Catch bugs in new features. Reduce the chances of screwing up old features
- Tests as documentation
- Testable code is well designed code
- Forces you to slow down and think.
- Defends against other programmers
- Fearless programming. Refactor!!
- For the fun of it - see code coverage reports and the green shade.

TDD

- Create tests first while development and bug fixing.
- Guard your functionality from being harmed.
- Solidify requirement

Best Practices

- Readable tests - Keep refactoring tests to make sure it is readable (Or soon you will have to start write super tests)
- Naming standards - function name and what this particular test is doing

Junit

- Java Unit Testing Framework
- Annotation Based - @BeforeClass, @Before, @Test, @After, @AfterClass
- Preconditions, function call, post asserts

Mockito

- Mocking framework that tastes really good and doesn't give you a hangover.
- Don't mock everything, then your tests are just mocks. Use mocking with great regret only if there is no other way.
- Don't mock fast and deterministic code in the same codebase
- Never mock value objects.
- Mocked objects
- when, thenReturn, thenThrow, Matchers - any(), eq()
- verify

Dependency Injection

- Wikipedia - dependency injection is a technique whereby one object supplies the dependencies of another object
- Abstractions should not depend on details. Details should depend on abstractions.
- Dependency injections helps in writing tests - mocks

Questions?