GENERALI

Department:

Country:

Test Driven Development (TDD)

Presentation Subtitle
Arial Regular 20/24pt

Agenda

- What is TDD?
- Benefits of TDD
- Conventional approach vs TDD approach
- TDD Best Practice

“No amount of testing can prove a software right, but a single test can prove a software wrong.”

- Amir Ghahrai

Software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved to pass the new tests, only.

Wikipedia

What is TDD?

What is TDD?

About “Design”
Not about testing…test first, then design code

Short development iterations
Forces constant integration eg. produces code to necessary pass test cases

Developers write tests
Gives confidence in code. Get direct feedback!

It’s automated!

Why don’t we do TDD?

Don’t know how to set up
Where can we find the tools and which tools will fit?

Tests are too complex or too deep
Some scenarios are too complicated or can’t be tested!

Too much to test, where to start?
Some large applications don’t have unit tests.

NOT ENOUGH TIME!!!

Benefits of TDD

Results

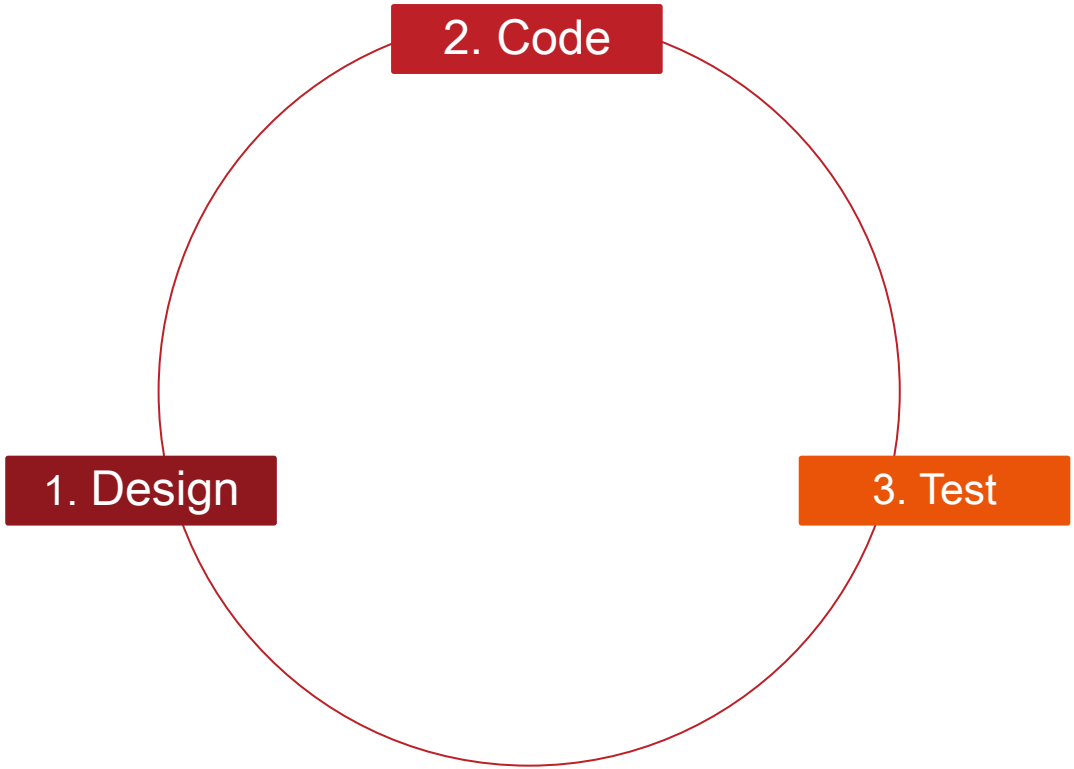
- Higher code quality
- Maintainability
- Flexibility
- Predictability

.....

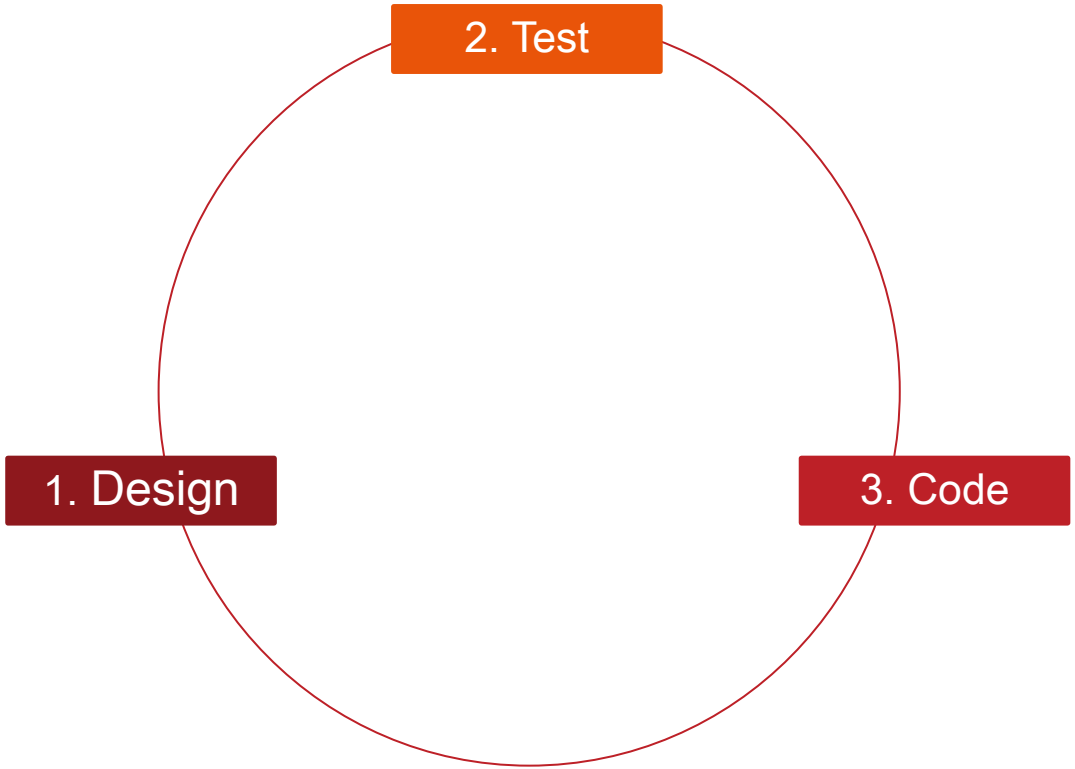
- A well designed application!

Conventional approach vs TDD approach

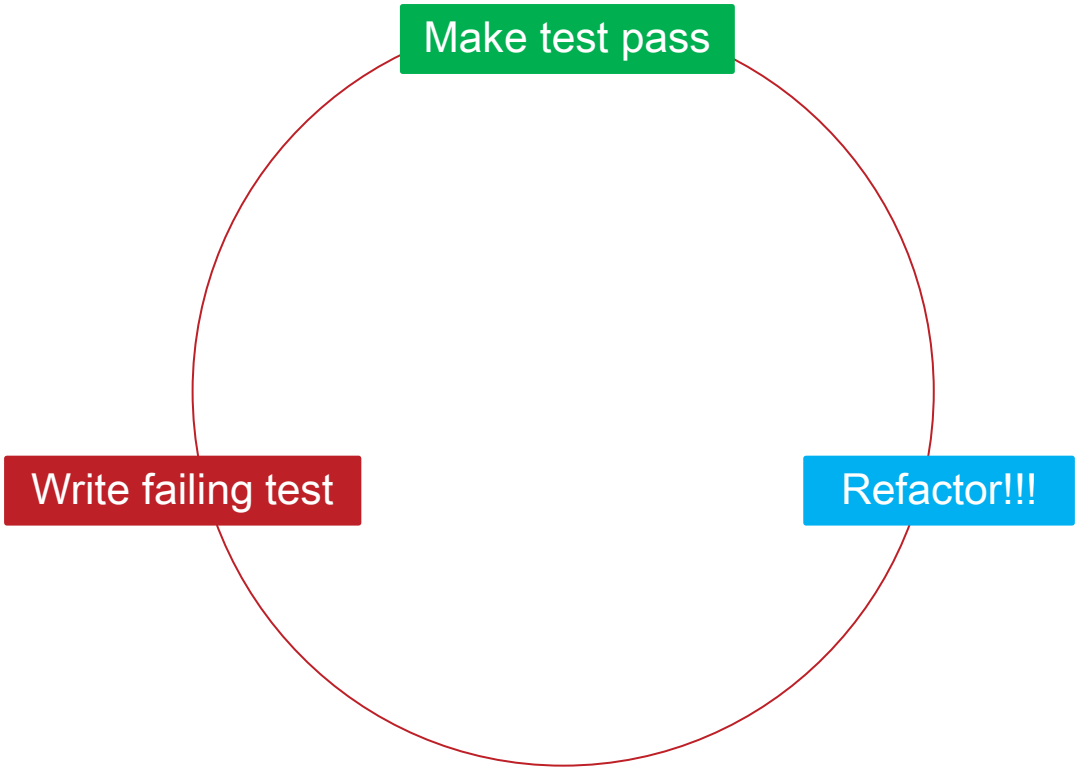
Conventional approach



New approach



TDD Cycle (Red-Green testing)



TDD Best Practice

TDD Principles

- What are you trying to do?
- Follow TDD Cycle
- Continually make small, incremental changes
- The build must work at all times

  Changes that break the system or tests that failures must be fixed!

Best Practices: Naming Convention

Naming Convention

- Help organize tests better
- Common understanding between teammates
- Easier for knowledge transfer

Descriptive Method Names

- Easier to figure out failed cases
- Facilitate checking for test coverages

GIVEN/WHEN/THEN Syntax

- Given – Precondition to the test (Not Mandatory)
- When – Testing condition
- Then – Expected test result

Best Practices: Naming Convention

[JAVA TEST]

```java
@Test
public final void whenOneNumberIsUsedThenReturnValueIsThatSameNumber() {
    Assert.assertEquals(3, StringCalculator.add("3"));
}

@Test
public final void whenTwoNumbersAreUsedThenReturnValueIsTheirSum() {
    Assert.assertEquals(3+6, StringCalculator.add("3,6"));
}
```

[JAVA IMPLEMENTATION]

```java
public static int add(final String numbers) {
    int returnValue = 0;
    String[] numbersArray = numbers.split(",");
    if (numbersArray.length > 2) {
        throw new RuntimeException("Up to 2 numbers separated by comma (,)
    }
    for (String number : numbersArray) {
        if (!number.trim().isEmpty()) { // After refactoring
            returnValue += Integer.parseInt(number);
        }
    }
    return returnValue;
}
```

Best Practices: Naming Convention

```java
package com.wordpress.technologyconversations.tddtest;

import org.junit.Test;
import com.wordpress.technologyconversations.tdd.StringCalculator;

public class StringCalculatorTest {
    @Test(expected = RuntimeException.class)
    public final void whenMoreThan2NumbersAreUsedThenExceptionIsThrown() {
        StringCalculator.add("1,2,3");
    }
    @Test
    public final void when2NumbersAreUsedThenNoExceptionIsThrown() {
        StringCalculator.add("1,2");
        Assert.assertTrue(true);
    }
    @Test(expected = RuntimeException.class)
    public final void whenNonNumberIsUsedThenExceptionIsThrown() {
        StringCalculator.add("1,X");
    }
}
```

Activity

Naming Practice

- You have 1 minute to think about a test case
- Use the pattern GIVEN/WHEN/THEN to build a sentence

Example:

- Scenario – Calculate the tax
- Test Case – Given a user earns $5000 salary, when tax rate is 20% then the tax is $1000

Best Practices: Processes

Write Test Before Implementation

- Ensures that testable code is written
- Ensures that every line of code gets tests
- Developer focus on requirements before coding

Write Code Only If Test Failed

- Ensure test does not work without implementation
- Ensure new function is needed for new requirement

Rerun All Tests Before Code Change

- Ensure no side-effect caused by code changes
- Only focus on new piece of code change
- Safe refactoring

Best Practices: Development

Write the simplest code to pass test

- Avoid unnecessary features

Write Assertion First, Code Later

- Clarifies the purpose of requirement earlier

No Dependencies between Tests

- Each test should be independent
- Dependency may cause false test result
- Reduce the complexity to reorganize test cases if requirement has significant change

Benefits

- Save time for retest
- Easy and safe refactoring
- Increase quality and test coverage
- Product better design

You still have to do…

Unit testing

Integration testing

UI / UX Testing

- Web Content Accessibility Guidelines (WCAG) 2.0

Regression

Questions?

Exercise 1 – Individual Exercise

Summary

- What is TDD?
- Benefits of TDD
- Conventional approach vs TDD approach
- TDD Best Practice

Thank You.

Contacts: