# Introduce Yourself

- Your name
- Why are you here?
- What do you expect to learn?

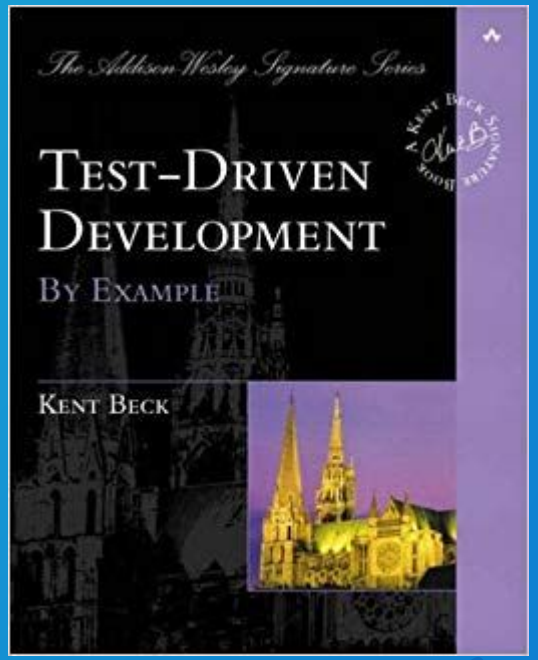
# Why do we need Coding Gym?

Become a better developer through deliberate exercises

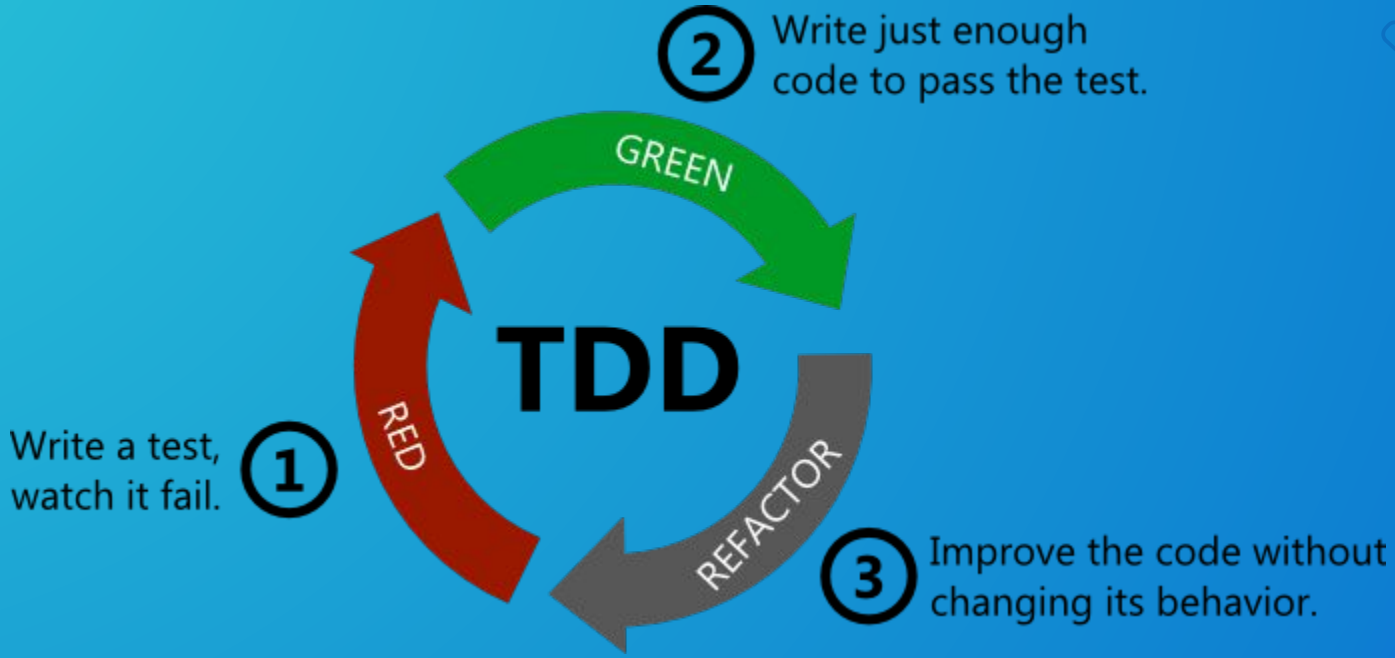
# Introduction to Test Driven Development

# Who “discovered” TDD?

# Kent Back


The Addison Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck

# What is it about?


2
Write just enough
code to pass the test.
GREEN
TDD
RED
REFACTOR
Write a test,
watch it fail.
1
3
Improve the code without
changing its behavior.

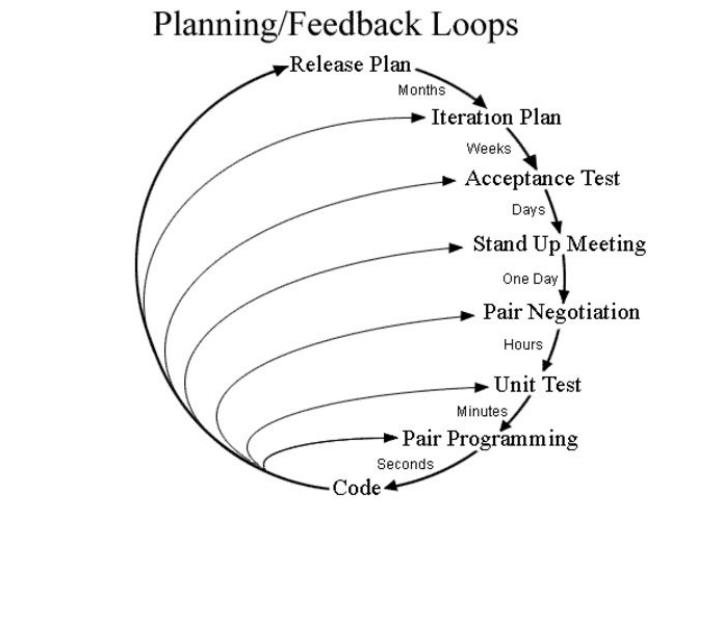
# What are the benefits?

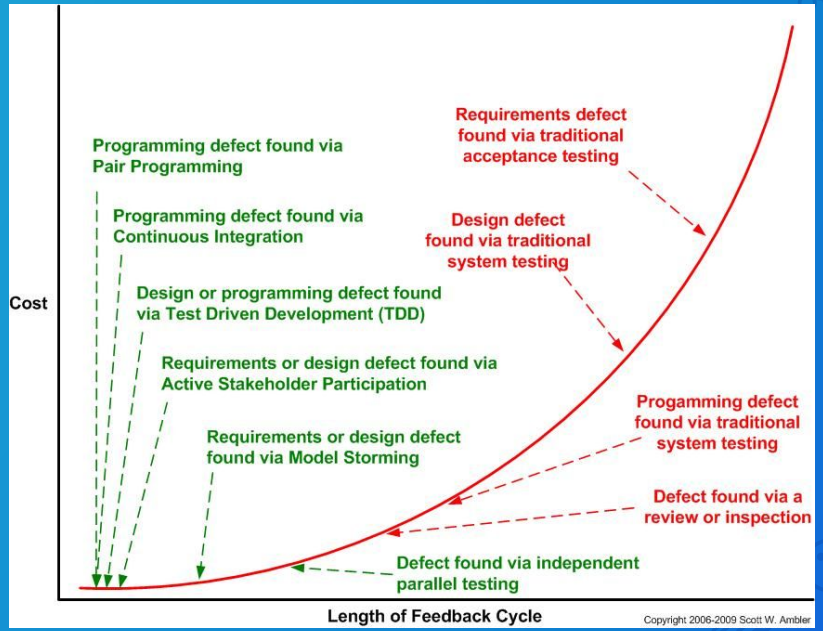
“

Clean codes that works

- Ron Jeffries

# Benefit #1: Faster Feedback



## Benefit #2: A safety net to protect you

- Existing tests help to catch regression errors when you make further changes

## Benefit #3: Focus on one task at a time

- One task at a time:
  - Writing a failing test
  - Make the failing test pass
  - Refactor
- Move in baby steps

## Benefit #4: Thinking from users' perspective

- Writing tests first forces you to think from the users' perspective
- This leads to better design
  - User friendly API
  - Smaller and modular units

## Benefit #5: Less waste

- You write tests to define how the system-under-test should be used
- You only write enough production code to make the tests pass

# First TDD Exercise

Implement a number sequence generator with TDD

# A Number Sequence

Here is a sequence of numbers with some pattern in it.

To start with, the first number in the sequence is 0

The second number in the sequence is 1

The next number is 1

The next number is 2

The next number is 3

.....This is Fibonacci number sequence!!!

# Fibonacci Sequence

The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Definition:

- First element is 0
- Second element is 1
- From 3rd element onwards: every element is the sum of previous two elements

# TDD Rules

# Rule #1

Start with a test.

Write new production code only if an automated test has failed

# Rule #2

Baby steps: only one failing test at a time

# Rule #3

Baby steps: write *small* test

## What's a small test?

- Each test should focus on a specific feature/behavior of the system under test
- It should not take you much time (< 10 mins) to make it pass
- It should only have one reason to fail
  - Implication: in many cases, there should only be one assertion in the test

# Rule #4

Baby steps: write enough production code to make the failing test pass. No extra code.

# Rule #5

Baby steps: implement the simplest algorithm first, then generalise it later when you identify some patterns

# Rule #6

Don't forget about Refactoring

# Rule #7

Refactor your tests too!

# Rule #8

Don't refactor when your tests are failing.
Make tests pass first.

# More on Baby Steps

# Why does it help?


SBS
hamilton

# Can we move in bigger steps?

Yes, under certain situations.

# Let’s do it again

Implement Fibonacci number generator with TDD

# Fibonacci Sequence

The Fibonacci Sequence is the series of numbers:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Definition:

- First element is 0
- Second element is 1
- From 3rd element onwards: every element is the sum of previous two elements

# Did you feel the difference?

# Lesson Learnt

**Move in smaller baby steps:**

- When you are new to the problem
- When you haven't found the pattern

**Move in bigger baby steps:**

- When you are familiar with the problem
- When the pattern is well known

# Reflection

# Did you have any surprises?

# Which part makes you feel uncomfortable?

# How long did it take you to make one test pass?

# Do I need to write tests for every function?

How about private/helper methods?

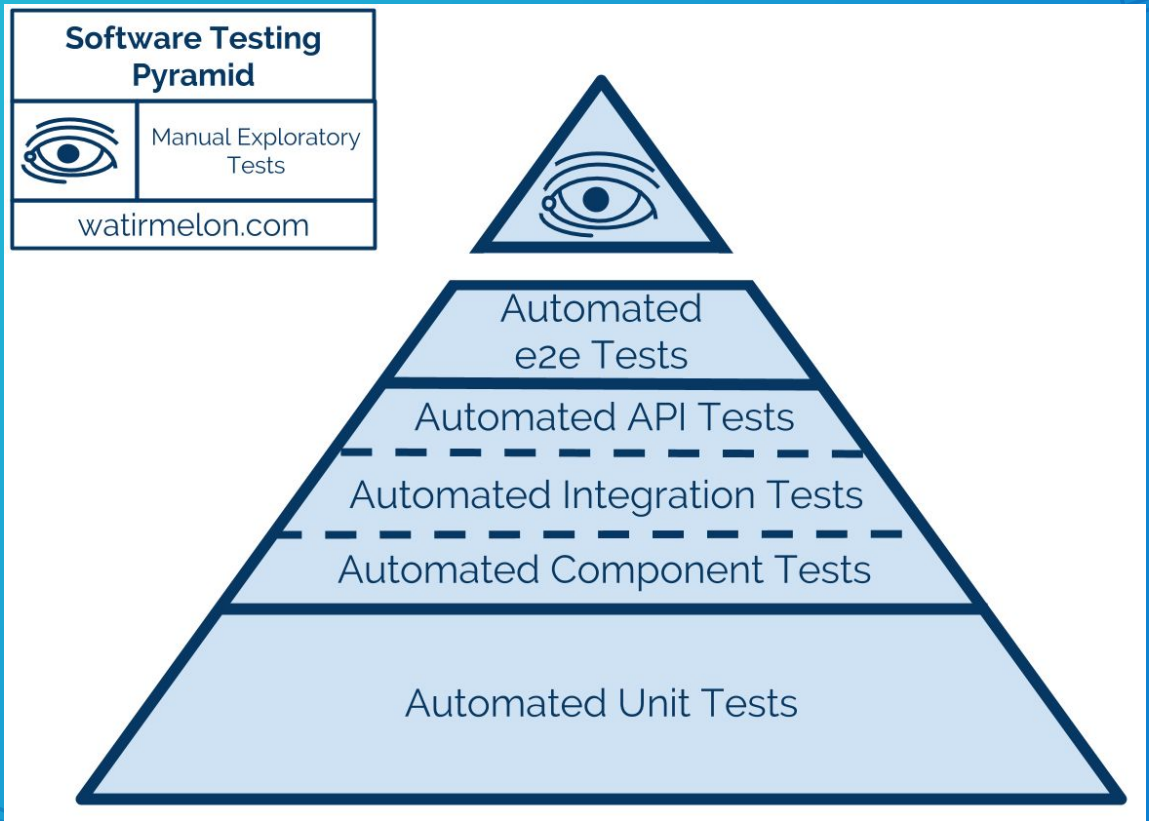
# Can we write tests later?

## Later == Never

- You have less interests to write test when your production code is already working
- You may write some codes that's never used
- You may write some codes that's hard to test/use

# What is the T in TDD?

What kind of tests? Unit Tests? Or Acceptance Tests? Or both?


Software Testing Pyramid
Manual Exploratory Tests
watirmelon.com
Automated e2e Tests
Automated API Tests
Automated Integration Tests
Automated Component Tests
Automated Unit Tests

# Good books on TDD