TDD

Workshop part 2

2017-03-30 | Valtech | Mikael Lundin | valte.ch/TDDWorkshop2

Schedule

Day 1

- 09-10 : Introduction to TDD
- 10-12 : Exercises
- 12-13 : Lunch
- 13-14 : Test-Driven Design
- 14-16 : Exercises

Test-Driven Design

or test-driven development as a design tool

or test-driven development process

Test-Driven Design

Improving testability

- Low Coupling
- High Cohesion

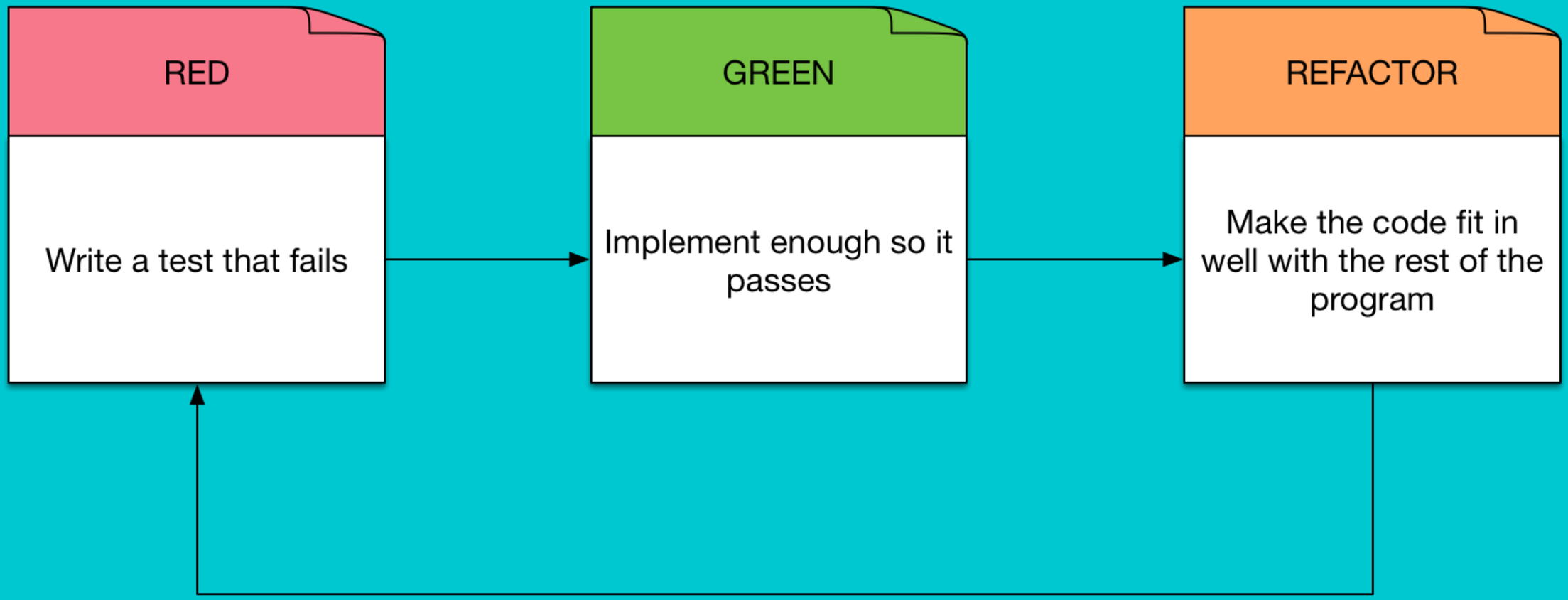
Test-First Development

- First write your test
- Then write the code

How do Test-first affect design?

- Less over-engineering
- Avoid the what-ifs

Red Green Refactor!



API design through tests

Writing code that uses the API is the best way to come up with a good API design.

```
var client = new SpamService("http://spam.api");


var users = client.GetAllUsers();
foreach (var user in users)
{
  client.SendSpam(user.Email);
}
```

Bug fixing test-first

BUG REPORT

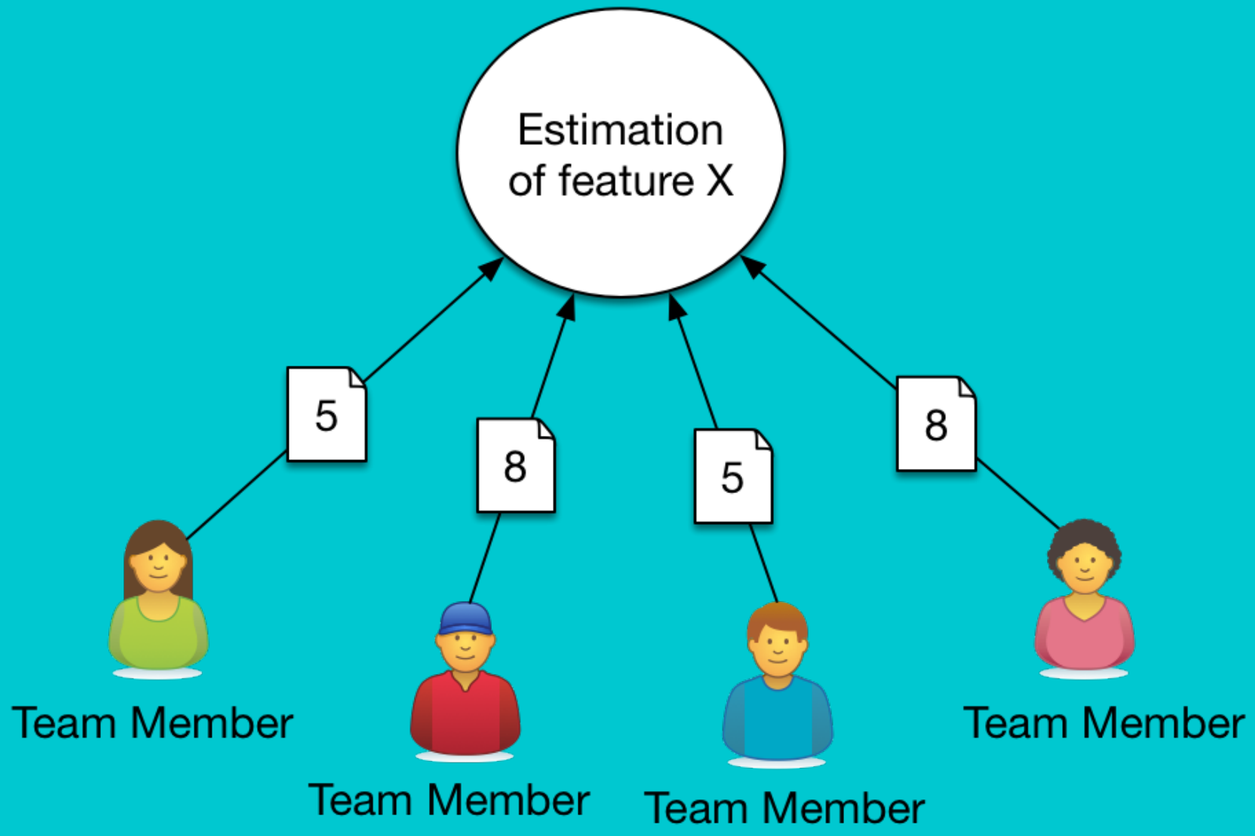
English translations do not turn up on the english news page.

Turn into the following test.

```
[Test]
public void Given_user_language_is_english_When_navigating_to_news_page_Then_headlines_should_be_in_english()
{
  //...
}
```

Exercises!

Poker Planning



Poker Planner Specification

1. Should start a new game by specifying number of players and play that game.
2. Once game is played, all played cards to be the result.
3. The game should support several players at once.
4. The game should not give result until all players have put down their card.

Exercise 2:1

A poker planning deck has the following cards 1, 2, 3, 5, 8, 13, 20, 40, 100 and Coffee. You play the Coffee card when you need a coffee break.

Use Red/Green/Refactor method to make sure that only valid cards can be played.

- Red: Write a test that fails
- Green: Make the test pass with least amount of implementation
- Refactor: Update the code to look good and work well

Exercise 2:2

You get a problem report from a user that you still are able to play a game that has been completed. Make sure that you cannot play completed games.

Formulate the problem as a test. It should be red. Now, fix the problem in your SUT and make sure the test turns green.

Exercise 2:3

You get a change request from a user that wants to be able to replay a game. Create new functionality that will restart a game and let you play it again.

Formulate the change as a test. It should be red. Make it green by least amount of implementation. Refactor it to work well with the rest of the program.