TDD

Tamás Ambrus, István Gansperger

Eötvös Loránd University
ambrus.thomas@gmail.com

Introduction

Why do we need to test?

We want to avoid several adverse occasions:

- putting or refactoring some functionality in a legacy code is hard,
- you can never know what you blew,
- reliability by users shrinks with every bug,
- you can never be sure that your code works.

A good solution - still not perfect

We test our code:

- to point out the defects and errors we made during the implementation (or requirements analysis, and so on),
- since it makes sure of the customer's reliability in the application,
- we want ensure the quality of the product,
- continuous testing makes bugs evanescent, while failures without 'tested background' can be very expensive.

Testing levels

There are many levels of testing for each development life cycle phase.
This course cares about the **smallest one: unit testing**. As this is the smallest unit in testing, it implies that unit tests should produce the biggest percentage of 'test code writing'.

As the name shows: they focus on units like methods or classes.

Unit test

Definition

A piece of code written by the developer that executes a specific functionality in the code to be tested without external dependencies. It asserts a certain behavior or state. The unit tests target the smallest unit of code, especially a function or a class.

In Java, we use the **JUnit framework** to write unit tests.

Initial example

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;
import org.junit.Before;

public class FibonacciTester {
  private Fibonacci f;

  @Before
  public void setUp() {
    f = new Fibonacci();
  }

  @Test
  public void firstThreeMatchToDefinition() {
    // assertEquals(the result that I should get, the result
    // that I actually got)
    assertEquals(0, f.fib(1)); // or assertTrue(0 == f.fib(1));
    assertEquals(1, f.fib(2));
    assertEquals(1, f.fib(3));
  }
}
```

Annotations everywhere

Here's a list about the annotations that will be preferred during this course:

- **@Test**: tells JUnit that the *public void* method to which it is attached can be run as a test case,
- **@Ignore**: sometimes you want to temporarily disable a test or a group of tests,
- **@Before**: when writing tests, it is common to find that several tests need similar objects created before they can run,
- **@BeforeClass**: sometimes several tests need to share computationally expensive setup (like logging into a database),
- **@After**: if you allocate external resources in a *@Before* method you need to release them after the test runs.

Let's summarize

What have we achieved? We have tests, after we wrote our lines of code.

- Is it good?
- Could we do it better?

TDD

Test-Driven Development

Definition

Test-driven development (TDD) is an evolutionary approach to development which combines test-first development where you write a test before you write just enough production code to fulfill that test and refactoring.

It is one way to think through your requirements or design before you write your functional code.

Uncle Bob's law (Robert C. Martin - biggest fan of TDD :>)

The three laws of TDD

1. Write NO production code except to pass a failing test.

Uncle Bob's law (Robert C. Martin - biggest fan of TDD :>)

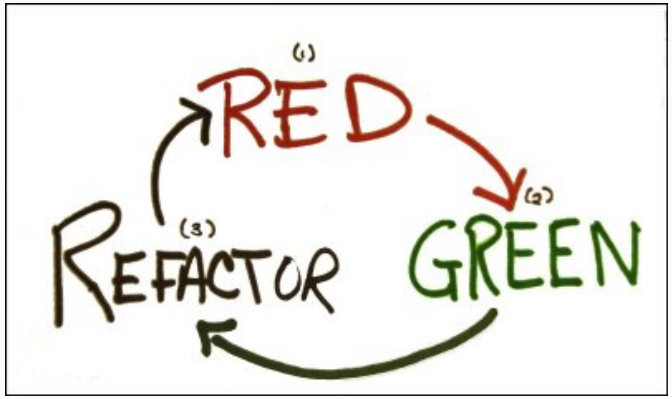
The three laws of TDD

1. Write NO production code except to pass a failing test.
2. Write only enough of a test to demonstrate a failure.

Uncle Bob's law (Robert C. Martin - biggest fan of TDD :>)

The three laws of TDD

1. Write NO production code except to pass a failing test.
2. Write only enough of a test to demonstrate a failure.
3. Write only enough production code to pass the test.



One last rule

Anyway, be careful with these rules. In general, they say that write only enough code that you need at the moment.
It is not enough. As a result you need to get a perfectly structured and refactored code that fulfills not only the tests.
Tests are only subsets of possible use cases. The production code must be written well for any possible test cases. Don't forget this.

References

- http://istqbexamcertification.com/why-is-testing-necessary/
- http://istqbexamcertification.com/what-are-software-testing-levels/
- http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd