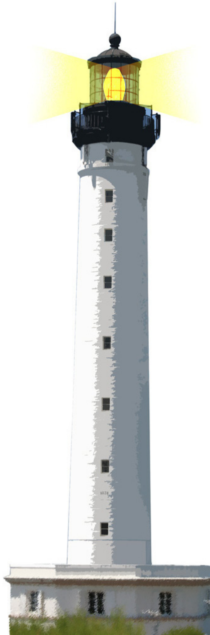
Learning Object-Oriented Programming and Design with TDD



Unit Testing in a Nutshell

Stéphane Ducasse
http://stephane.ducasse.free.fr

http://www.pharo.org

W1S04

Why testing

- Specify
- Verify
- Support evolution

A Test

In a test, we

- Create a context: Create an empty set
- Send a stimulus: Add twice the same element
- Check the results: Check that the set contains only one element

Testing set addition

```
set1 := Set new.
set1 add: 1.
set1 add: 2.
set1 add: 1.
set2 size = 2.
>>> true
```

Testing set addition

```
set1 := Set new. "Context"
set1 add: 1. "Stimulus"
set1 add: 2. "Stimulus"
set1 add: 1. "Stimulus"
set2 size = 2. "Verification/Assertion"
>>> true
```

To provide a frame to define and execute tests

With SUnit:

- You define special class
- You define special methods

We will come back later to this.

(SetTestCase)

```
TestCase subclass: # SetTestCase
  ...
```

```
SetTestCase >> testAdd
 | empty |
 empty := Set new.   "Context"
 empty add: 1.   "Stimulus"
 empty add: 2.
 empty add: 1.
 self assert: empty size = 2.   "Check"

SetTestCase run: #testAdd
```

(In a Subclass of TestCase)

Each method starting with `test*`:

- Represents a test
- Is automatically executed

The results of the test are collected in a `TestResult` object

Why testing is important: Specify

Your tests are your first clients

- Help you to design nice API
- Make sure that you control your functionality

Why testing is important: Verify

Your tests are your insurance (regression testing)

- When adding a new behavior, control that there is no side effect and bug introduction in existing code

Why testing is important: Support evolution

Changes are

- Inevitable
- Programs represent the world and world is changing!
- Better be prepared

When you have to change your existing code:

- Identify side effects
- Validate that changes do not break more than they should do

Unit testing

- Simple
- Focus on one action

There are other testing approach

- Integration testing
- Acceptance testing...

Summary

- Unit tests are easy to create and run
- Create one test and run it million times!
- Use them as your life insurance
- Libraries exist (BabyMock, Mocketry) for different styles of testing

Resources

- Pharo Mooc - W5S06 Videos
- Pharo by Example `http://books.pharo.org`

A course by Stéphane Ducasse
`http://stephane.ducasse.free.fr`

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
`http://mooc.pharo.org`