Automatic Refactoring Implementation using TDD & Meta-Programming

Hernán Wilkinson

Twitter: @HernanWilkinson
www.10pines.com
Classes: university.10pines.com
Goals

- Learn What is and How is implemented an Automatic Refactoring
- Learn Meta-Programming
- Practice TDD with a uncommon problem
- Do it in a Live Coding Environment!
Organization

- Learn Smalltalk – 1hr
- Learn Meta-Programming – 30 min
- Break
- Refactoring Implementation – 1 hr.
- Conclusion & Unexpected: The rest 😊
Material

- tinyurl.com/XP2018Meta-Workshop
  - One zip per OS (linux64, macos64 and windows 64)
  - Slides

- The zip includes:
  - Development Environment
  - Solution step by step
Smalltalk

- Alan Kay
- Dan Ingalls
- Adele Goldberg
Smalltalk

Augment Children Comprehension

Smalltalk (72, 74, 76, 78, 80)

Object Oriented

GUI - IDE

VM

DynaBook

Simula 67

Lisp

Flex Machine

http://www.youtube.com/watch?v=AuXCC7WSczM
Why Smalltalk?
Let’s start!

▶ Execute run.sh or run.bat
▶ Stop on Tools-Browsing
Meta-Programming
Meta-Programming

- Meta-xxx: That talks about or defines xxx
- Example:
  - A class is a meta-instance
  - English/Spanish/etc are meta-languages
(Reflection and Open Implementations - Eric Tanter)
Meta-Programming

- Reflexión:
  - Hability to operate, represent and treat itself the same way it operates over other subject
- Introspection = Read
- Intercession = Write
- Structural
- Behavioral
# Meta-Programming

<table>
<thead>
<tr>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure</strong></td>
<td><strong>Automatic Refactorings</strong></td>
</tr>
<tr>
<td>- Coding statistics (lines of code, etc.)</td>
<td>- Automatic patterns implementation</td>
</tr>
<tr>
<td>- Design rules</td>
<td></td>
</tr>
<tr>
<td><strong>Behavior</strong></td>
<td><strong>Create Method</strong></td>
</tr>
<tr>
<td>- Assertion Name</td>
<td>- Pluggable Proxy</td>
</tr>
<tr>
<td>- Debugger</td>
<td>- Debugger</td>
</tr>
</tbody>
</table>

Meta-Programming
Meta-Circular Languages
Let's start!

From Meta-Programming to Behavioral-Write
TDD
What TDD is

- A learning technique
  - Iterative
  - Incremental
  - With Immediate Feedback

- Side effects
  - Remembers all we learned
  - Tells us as soon we make a mistake
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - Must fail the first time it is run
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - Must fail the first time it is run

2) Run all tests
   - Implement the simplest solution that will make the test/s pass
   - GOTO 2 until all test pass
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - Must fail the first time it is run

2) Run all tests
   - Implement the simplest solution that will make the test/s pass
   - GOTO 2 until all tests pass

3) Reflect – Can we have a better impl./design?
   - Yes -> Refactor. GOTO 2
   - No -> GOTO 1
How to do TDD?

1) Write a test
   - It has to be **the simplest** one
   - Must fail the first time it is run

2) Run all tests
   - Implement the simplest solution that will make the test/s pass
   - GOTO 2 until all tests pass

3) Reflect – Can we have a better impl./design?
   - Yes -> Refactor. GOTO 2
   - No -> GOTO 1
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - **Must fail** the first time it is run

2) Run all tests
   - Implement the simplest solution that will make the test/s pass
   - GOTO 2 until all tests pass

3) Reflect – Can we have a better impl./design?
   - Yes -> Refactor. GOTO 2
   - No -> GOTO 1
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - Must fail the first time it is run

2) Run all tests
   - Implement the simples solution that will make the test/s pass
   - GOTO 2 until all test pass

3) Reflect – Can we have a better impl./design?
   - Yes -> Refactor. GOTO 2
   - No -> GOTO 1
How to do TDD?

1) Write a test
   - It has to be the simplest one
   - Must fail the first time it is run
2) Run all tests
   - Implement the simplest solution that will make the test/s pass
   - GOTO 2 until all tests pass
3) Reflect – Can we have a better impl./design?
   - Yes -> Refactor. GOTO 2
   - No -> GOTO 1
Test structure

**Setup**
- Creates the test’s context

**Exercise**
- Exercises the functionality that is being tested. Defines WHAT is being tested

**Assert**
- Verifies that happened what it was expected to happen or fails otherwise
Let's start!

▶ From Tools-TDD to Exercise
Conclusions
Thanks!

info@10pines.com
www.10Pines.com
University.10pines.com

twitter: @10Pines
Twitter: @hernanwilkinson

Argentina

Tel.: +54 (11) 6091-3125
Av. Alem 896, 6to
(1001) Buenos Aires