Learning Object-Oriented Programming, Design and TDD with Pharo

Stéphane Ducasse

March 11, 2019

Layout and typography based on the `sbabook` LaTeX class by Damien Pollet.

Contents

Illustrations

CHAPTER 1

About this book

1.1 A word of presentation

I started to write this book back in 1998 when I wrote around 900 pages in preparation for *Learning Programming with Robots* (Apparently, I needed to write to understand what I wanted to explain and how). From this I extracted *Learning Programming with Robots*, which was a book to teach simple concepts such as variables, loops, procedures and to help people teach kids how to program. My original objective was to write a second volume to teach object-oriented programming. But while this first volume was a success, I got really frustrated because to be understandable by everyone I had to remove what I like: object-oriented programming and good object-oriented design.

At that time, I met Harald Wertz, who gave me really nice ideas and pointers such as L-systems, then asked why I focused on procedural thinking and suggested that I should teach object-oriented programming instead. And he was right. This remark was like a bee in my bonnet for more than ten years. In fact, it was my original objective but I was exhausted after my first attempt and I had to focus on my academic life.

Now, nearly fifteen years later, I'm ready to write a book to start with object-oriented programming. In fact, I rewrote everything I got from that time. I hope that you will enjoy it as much as I did — even if, for me, writing a book is a really long and daunting task because I want to make it great. I plan to write another volume on patterns of design that will extend this book.

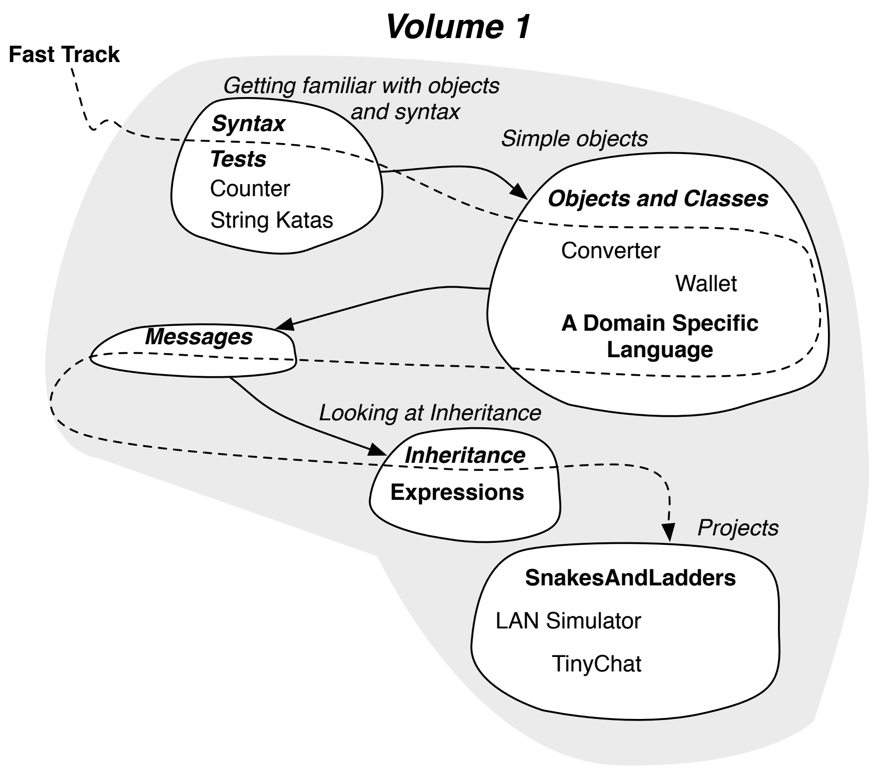

Figure 1-1 Reading maps.

1.2 Structure of the book

While writing this book, I faced a challenge to find the correct level of difficulty. To solve this problem, I structured the book either into key chapters on basic concepts, or into projects on more advanced topics. The projects are little tutorials or more realistic examples, with step by step explanations; you can skip over them and come back to read them whenever you feel like it. I also propose various paths through the book with different levels of reading; however, many of the *simpler* chapters also contain design remarks.

Fast track

The following chapters contain more conceptual information:

In the volume 1:

- Glimpse of the syntax
- Tests, tests and tests

- Objects and classes
- Revisiting objects and classes
- Domain specific language
- Inheritance and expressions
- Sending messages
- Snakes and ladders

The other chapters are more exercise and practical. For example, with Tiny-Chat, you will have fun with a web server written in the single page of code. You will find the solutions of the exercises in a separate pdf available on the book web site at http://books.pharo.org and the associated github repository https://github.com/SquareBracketAssociates/LearningOOPWithPharo.

1.3 What you will learn

I would like to present the concepts that I want to teach you and that hopefully you should acquire. What is key to understand is that I will focus on the *key* conceptual elements. It is easy for me because I will not explain OOP/D in general but within the context of Pharo and Pharo is the essence of Object-Oriented programming since its object model is minimal but it covers the key and essential aspect of OOP. For example, we will not present method modifiers, types, or overloading (which is a bad concept).

We will focus on object-oriented *programming* concepts:

- Objects / Classes
- Messages / Methods
- `self` and its semantics
- Inheritance
- `super` and its semantics

...and on object-oriented *design* concepts:

- Class responsibility collaboration
- Delegation
- Message sends are choice
- Message sends are plans for reuse
- The ”Don’t ask, tell” Principle
- Tests are your life insurance
- Polymorphism

In addition, we will also present

- Tests
- Software refactorings

Growing software

Often books present a problem and its solution. Now for non-trivial problems, the solution does not fall from the sky or get developed in one stroke, but it is the constant evolution of a first solution that evolves over time. Such an evolution is often difficult and tedious because the developer jumps from one stable state to a situation where his code may not work anymore. This is where Test Driven Design and refactorings really help. Test Driven Design helps focusing on new features and captures them as executable entities: tests. Refactorings help by transforming code without breaking its invariants. Note that tests do not forbid to break code, they help identifying when previous invariants or constraints got violated. Sometimes violated tests identify a bug, but they may be broken just because the requirements changed and that the tests should be updated. In this book, I wanted to see how software grows in little steps. This is what I do frequently during my coding sessions, and I think that this is important, to cover the hidden paths in software creation.

Syntax, blocks and iterators

Since we need a language to express our programs, we will teach you the syntax of Pharo. In particular, we will use some simple chapters to get you started.

Now in a nutshell, you should know that the Pharo syntax

- fits in one postcard and
- is based on objects, messages and closures.

Note that closures are not a recent addition to the language but a central cornerstone. Closures are the foundation for conditionals and loops. They enable this ’messages all over the place’ syntax as well as really powerful iterators.

1.4 Typographic conventions

Pharo expressions or code snippets are represented either in the text as `'Hello'` and `'Hello' reversed`, or for more substantial snippets, as follows:

```
'Hello'
```

When we want to show the result of evaluating an expression, we show the result after three chevrons >>> on the next line, like so:

```
'Hello' reversed
>>> 'olleH'
```

Whenever we feel the text makes a point that is important or technical enough to be highlighted, we will do so with a thick bar:

> **Important** This is a point that is worth drawing some more attention.

Finally, the coffee cups highlight some points to take away and serve as a concise summary of the sections:

☕ If you skim through a section, take a few seconds to check for coffee cups!

1.5 Videos

While reading this book, you can also use some of the videos produced for the Pharo mooc. All the videos are available at http://mooc.pharo.org. I strongly suggest to watch the videos explaining how to use and interact with the environment.

1.6 Thanks

I would like to thanks Morgane Pigny, Anne Etien, Quentin Ducasse, Sven van Caekenberghe, Hayatou Oumarou, Kateryna Aloshkina, Ricardo Pacheco, Olivier Auverlot, Mariette Biernacki, Herby Vojcik, Denis Kudriashov, Holger Freyther, Dimitris Chloupis, Amal Noussi, René Paul Mages, Hannes Hirsel, Lorenzo Solano Martinez for their great feedback. Alexandre Bergel for his examples on messages. Olivier Auverlot for his constant enthousiam and for TinyChat. Guillermo Polito for the idea of file and directory example. Damien Pollet for this great template and the new LAN implementation and the numerous makefile implementation and Pillar help.

Bibliography