

Learning Object-Oriented Programming, Design and TDD with Pharos

Stéphane Ducasse

March 11, 2019

Copyright 2017 by Stéphane Ducasse.

The contents of this book are protected under the Creative Commons Attribution-ShareAlike 3.0 Unported license.

You are **free**:

- to **Share**: to copy, distribute and transmit the work,
- to **Remix**: to adapt the work,

Under the following conditions:

Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page:
<http://creativecommons.org/licenses/by-sa/3.0/>

Any of the above conditions can be waived if you get permission from the copyright holder. Nothing in this license impairs or restricts the author's moral rights.



Your fair dealing and other rights are in no way affected by the above. This is a human-readable summary of the Legal Code (the full license):
<http://creativecommons.org/licenses/by-sa/3.0/legalcode>

Contents

Illustrations	ii
1 Summing and converting money	1
1.1 Requirements	1
1.2 Given context	1
1.3 Solution	2
Bibliography	3

Illustrations

Summing and converting money

We will now work on one example proposed by A. Bergel and we would like to thank him for it.

```
[ 1 EUR = 662 CLP (Chilean pesos)
```

1.1 Requirements

```
[ TestCase subclass: #CurrencyTest
  CurrencyTest >> testSum
    | clp1 eur1 clp2 eur2 |
    clp1 := CLP new value: 3500.
    eur1 := EUR new value: 10.
    clp2 := CLP new value: 5000.
    eur2 := EUR new value: 20.

    self assert: clp1 + clp2 equals: (CLP new value: 8500).
    self assert: clp1 + eur1 equals: (CLP new value: 3500 + 6620).

    self assert: eur1 + eur2 equals: (EUR new value: 30).
    self assert: eur1 + clp2 equals: (EUR new value: 5000 / 662 + 10).
```

In addition in a second step we will add conversion between Euros and USD.

1.2 Given context

You have a class `Currency` to which you can sum other `currencyCurrency`.

```
[ Object subclass: #Currency
  instVarNames: ''value
```

```
[ Currency >> + anotherCurrency
  self subclassResponsibility
[ Currency >> printOn: str
  super printOn: str.
  str nextPut: $<.
  str nextPutAll: self value asString.
  str nextPut: $>.
```

1.3 Solution

```
[ Currency >> sumWithEUR: money
  self subclassResponsibility
[ Currency >> sumWithCLP: money
  self subclassResponsibility
[ Currency >> = anotherCurrency
  ^ self class == anotherCurrency class and: [ self value =
    anotherCurrency value ]
```

You have two subclasses:

```
[ Currency subclass: #EUR
[ Currency subclass: #CLP
[ EUR >> + anotherCurrency
  ^ anotherCurrency sumWithEUR: self
[ EUR >> sumWithEUR: money
  ^ EUR new value: self value + money value
[ EUR >> sumWithCLP: money
  ^ CLP new value: (self value * 662) + money value
[ CLP >> + anotherCurrency
  ^ anotherCurrency sumWithCLP: self
[ CLP >> sumWithEUR: money
  ^ EUR new value: (self value / 662) + money value
[ CLP >> sumWithCLP: money
  ^ CLP new value: self value + money value
```

Bibliography