

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 Electronic wallet solution	1
1.1 Using a bag for a wallet	1
1.2 Testing money	1
1.3 Checking to pay an amount	2
1.4 Biggest coin	2
1.5 Biggest below a value	2
1.6 Improving the API	2
1.7 Coins for paying: First version	2
Bibliography	5

Illustrations

Electronic wallet solution

Here are the possible solutions of the implementation we asked for the Wallet Chapter ??.

1.1 Using a bag for a wallet

```
Wallet >> add: anInteger coinsOfValue: aCoin
    "Add to the receiver, anInteger times a coin of value aNumber"

    bagCoins add: aCoin withOccurrences: anInteger
```

We can add elements one by one to a bag using the message `add:` or specifying the number of occurrences of the element using the message `add:withOccurrences:`.

```
Wallet >> coinsOfValue: aNumber

    ^ bagCoins occurrencesOf: aNumber
```

1.2 Testing money

```
Wallet >> money
    "Return the value of the receiver by summing its constituents"
    | money |
    money := 0.
    bagCoins doWithOccurrences:
        [ :elem : occurrence |
            money := money + ( elem * occurrence ) ].
    ^ money
```

1.3 Checking to pay an amount

```

Wallet >> canPay: amountOfMoney
  "returns true when we can pay the amount of money"
  ^ self money >= amountOfMoney

```

1.4 Biggest coin

```

Wallet >> biggest
  "Returns the biggest coin with a value below anAmount. For
   example, {(3 -> 0.5) . (3 -> 0.2) . (5-> 0.1)} biggest -> 0.5"

  ^ bagCoins sortedElements last key

```

1.5 Biggest below a value

```

Wallet >> biggestBelow: anAmount
  "Returns the biggest coin with a value below anAmount. For
   example, {(3 -> 0.5) . (3 -> 0.2) . (5-> 0.1)} biggestBelow:
   0.40 -> 0.2"

  bagCoins doWithOccurrences: [ :elem :occurrences |
    anAmount > elem ifTrue: [ ^ elem ] ].
  ^ 0

```

1.6 Improving the API

```

Wallet >> addCoin: aNumber
  "Add to the receiver a coin of value aNumber"

  bagCoins add: aNumber withOccurrences: 1

Wallet >> removeCoin: aNumber
  "Remove from the receiver a coin of value aNumber"

  bagCoins remove: aNumber ifAbsent: []

```

1.7 Coins for paying: First version

```

Wallet >> coinsFor: aValue
  "Returns a wallet with the largest coins to pay a certain amount
   and an empty wallet if this is not possible"
  | res |
  res := self class new.

```

1.7 Coins for paying: First version

```

    ^ (self canPay: aValue)
      ifFalse: [ res ]
      ifTrue: [ self coinsFor: aValue into2: res ]
  ]

  Wallet >> coinsFor: aValue into2: accuWallet
    | accu |
    [ accu := accuWallet money.
      accu < aValue ]
      whileTrue: [
        | big |
        big := self biggest.
        [ big > ((aValue - accu) roundUpTo: 0.1) ]
          whileTrue: [ big := self biggestBelow: big ].
        self removeCoin: big.
        accuWallet addCoin: big ].
    ^ accuWallet

```


Bibliography