

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 Challenge yourself	1
1.1 Challenge: Message identification	1
1.2 Challenge: Literal objects	3
1.3 Challenge: Kind of messages	3
1.4 Challenge: Results	4
1.5 Challenge: unneeded parentheses	5
Bibliography	7

Illustrations



Challenge yourself

In Pharo everything is an object and most computation happens by sending *messages* to objects. In this chapter we propose a list of exercises to challenge you with the syntax.

1.1 Challenge: Message identification

For each of the expressions below, fill in the answers:

- What is the receiver object?
- What is the message selector?
- What is/are the argument (s)?
- What is the result returned by this expression execution?

```
3 + 4
```

```
receiver:  
selector:  
arguments:  
result:
```

```
Date today
```

```
receiver:  
selector:  
arguments:  
result:
```

```

[ #('' 'World') at: 1 put: 'Hello'

    receiver:
    selector:
    arguments:
    result:

[ #(1 22 333) at: 2

    receiver:
    selector:
    arguments:
    result:

[ #(2 33 -4 67) collect: [ :each | each abs ]

    receiver:
    selector:
    arguments:
    result:

[ 25 @ 50

    receiver:
    selector:
    arguments:
    result:

[ SmallInteger maxVal

    receiver:
    selector:
    arguments:
    result:

[ #(a b c d e f) includesAll: #(f d b)

    receiver:
    selector:
    arguments:
    result:

[ true | false

    receiver:
    selector:
    arguments:
    result:

```

```
[ Point selectors
  receiver:
  selector:
  arguments:
  result:
```

1.2 Challenge: Literal objects

What kind of object does the following literal expressions refer to? It is the same as asking what is the result of sending the `class` message to such expressions.

```
[ 1.3
>
[ #node1
>
[ #(2 33 4)
>
[ 'Hello, Dave'
>
[ [ :each | each scale: 1.5 ]
>
[ $A
>
[ true
>
[ 1
>
```

1.3 Challenge: Kind of messages

Examine the following messages and report if the message is unary, binary or keyword-based.

```

[ 1 log
  >
[ Browser open
  >
[ 2 raisedTo: 5
  >
[ 'hello', 'world'
  >
[ 10@20
  >
[ point1 x
  >
[ point1 distanceFrom: point2
  >

```

1.4 Challenge: Results

Examine the following expressions. What is the value returned by the execution of the following expressions?

```

[ 1 + 3 negated
  >
[ 1 + (3 negated)
  >
[ 2 raisedTo: 3 + 2
  >
[ | anArray |
  anArray := #('first' 'second' 'third' 'fourth').
  anArray at: 2
  >

```


1.5 Challenge: unneeded parentheses

```
[ #(2 3 -10 3) collect: [ :each | each * each]
>

[ 6 + 4 / 2
>

[ 2 negated raisedTo: 3 + 2
>

[ #(a b c d e f) includesAll: #(f d b)
>
```

1.5 Challenge: unneeded parentheses

Putting more parentheses than necessary is a good way to get started. Such practice however leads to less readable expressions. Rewrite the following expressions using the least number of parentheses.

```
[ x between: (pt1 x) and: (pt2 y)
...

[ ((#(a b c d e f) asSet) intersection: (#(f d b) asSet))
...

[ (x isZero)
  ifTrue: [....]
(x includes: y)
  ifTrue: [....]
...

[ (OrderedCollection new)
  add: 56;
  add: 33;
  yourself
:]
```

```

[ ...
  ((3 + 4) + (2 * 2) + (2 * 3))
  ...
  (Integer primesUpTo: 64) sum
  ...
  ('http://www.pharo.org' asUrl) retrieveContents
  ...

```

Bibliography