



**Learning Object-Oriented
Programming and Design with TDD**

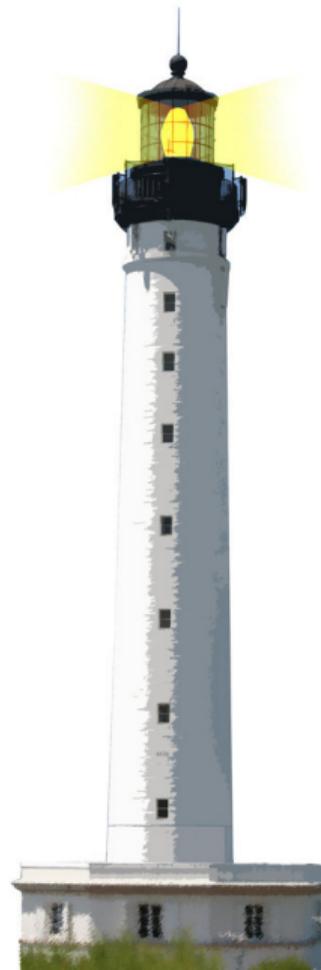
First Look at Class and Method Definitions

Stéphane Ducasse

<http://stephane.ducasse.free.fr>



W1S10



Class and Method Definitions in Pharo

- We will have specific lectures on classes and methods.
- Now this is just to give you a first impression
- Most of the time you will define classes and methods using tools
- There is no dedicated syntax for class definition just a message



Class Definition in Pharo

The screenshot shows the Pharo IDE interface for the `Point` class. The window title is "Point". On the left, there is a "Scoped Variables" pane with a tree view showing the package structure. The "BasicObjects" package is selected, and the "Point" class is highlighted. Below the tree view are buttons for "Hier.", "Class", and "Com.". The "History Navigator" pane on the right shows a list of categories: -- all --, accessing, arithmetic, comparing, converting, copying, extent functions, geometry, interpolating, point functions, polar coordinates, and printing. The main editor area displays the class definition for `Point`:

```
Object subclass: #Point
  instanceVariableNames: 'x y'
  classVariableNames: ''
  category: 'Kernel-BasicObjects'
```

Class Definition is a Message

```
Object subclass: #Point
  instanceVariableNames: 'x y'
  classVariableNames: ''
  package: 'Graphics'
```

We send the message `subclass:inst...` to the superclass to create the class



Method Definition in Pharo

The screenshot shows the Pharo IDE interface with the following components:

- Window Title:** Integer>>#factorial
- Buttons:** Scoped, Variables, History Navigator (with left and right arrows).
- Left Panel (Type: Pkg1|^Pkg2|P):**
 - Numbers (selected)
 - Objects
 - Pragmas
 - Processes
 - Protocols
 - Kernel-Tests
 - Keymapping-Cor
 - Keymapping-Key
- Middle Panel (Class/Method List):**
 - ExactFloatPrintPolicy
 - FloatPrintPolicy
 - InexactFloatPrintPolicy
 - Magnitude
 - Number
 - Float
 - Fraction
 - ScaledDecimal
 - Integer (selected)
- Right Panel (History Navigator):**
 - accessing
 - arithmetic
 - benchmarks
 - bit manipulation
 - comparing
 - converting
 - converting-arrays
 - enumerating
 - filter streaming
 - mathematical func (selected)
- Bottom Panel (Method Definition):**
 - factorial**
 - "Answer the factorial of the receiver."
 - `self = 0 ifTrue: [^ 1].`
 - `self > 0 ifTrue: [^ self * (self - 1) factorial].`
 - `self error: 'Not valid for negative integers'`

Method Definition in Pharo

factorial

"Answer the factorial of the receiver."

self = 0 ifTrue: [^ 1].

self > 0 ifTrue: [^ self * (self - 1) factorial].

self error: 'Not valid for negative integers'

In which class is factorial defined?

Presentation Convention

In this lecture, a method will be displayed as

```
Integer >> factorial
  "Answer the factorial of the receiver."
  self = 0 if True: [ ^ 1 ].
  self > 0 if True: [ ^ self * (self - 1) factorial ].
  self error: 'Not valid for negative integers'
```

- **Integer >>** is not part of the syntax
 - it tells you the method's class

Presentation Convention

The screenshot shows the documentation for the `Integer` class's `factorial` method. The interface includes a left sidebar with a class hierarchy, a central history navigator, and a main content area with the method's description and source code.

Integer>>#factorial

Scoped Variables History Navigator

Type: Pkg1|^Pkg2|Pl

- Numbers
- Objects
- Pragmas
- Processes
- Protocols
- Kernel-Tests
- Keymapping-Cor
- Keymapping-Key

- ExactFloatPrintPolicy
- FloatPrintPolicy
- InexactFloatPrintPolicy
- Magnitude
- Number
- Float
- Fraction
- ScaledDecimal
- Integer

- accessing
- arithmetic
- benchmarks
- bit manipulation
- comparing
- converting
- converting-arrays
- enumerating
- filter streaming
- mathematical func

- factorial
- gcd:
- lcm:
- nthRoot:
- nthRootRounded:
- nthRootTruncated:
- raisedTo:modulo:
- raisedToInteger:modulo:
- sqrt
- take:

Hier. Class Com.

factorial

"Answer the factorial of the receiver."

```
self = 0 ifTrue: [^ 1].  
self > 0 ifTrue: [^ self * (self - 1) factorial].  
self error: 'Not valid for negative integers'
```

1/6 [1] Format as you read W +L

Remember Messages

```
Integer >> factorial
```

```
"Answer the factorial of the receiver."
```

```
self = 0 ifTrue: [ ^ 1 ].
```

```
self > 0 ifTrue: [ ^ self * (self - 1) factorial ].
```

```
self error: 'Not valid for negative integers'
```

- factorial is the method name
- =, >, * and - are binary messages
- factorial is an unary message
- ifTrue: and error: are keyword messages
- the caret ^ is for returning a value

A Method Returns self by Default

```
Game >> initializePlayers  
self players  
at: 'tileAction'  
put: ( MITileAction director: self )
```

is equivalent to

```
Game >> initializePlayers  
self players  
at: 'tileAction'  
put: ( MITileAction director: self ).  
^ self    "<--- optional"
```



Class Methods

The screenshot shows the Smalltalk IDE interface for editing the `Point` class. The title bar reads "Point class>>#x,y:". The interface is divided into several sections:

- Class Browser:** Located on the left, it shows a hierarchy of classes including `Character`, `CombinedChar`, `Margin`, `Point`, and `Rectangle`. The `Point` class is currently selected.
- History Navigator:** Located at the top right, it shows a list of recent actions, including "instance creation" and "*System-Setting".
- Code Editor:** The main area displays the class definition for `Point`. The code is as follows:

```
x: xInteger y: yInteger
"Answer an instance of me with coordinates xInteger and yInteger."

^ self basicNew setX: xInteger setY: yInteger
```

- press the button class to define a class method
- in lectures, we add class

`Point` class >> x: xInteger y: yInteger

"Answer an instance of me with coordinates xInteger and yInteger."

`^ self basicNew setX: xInteger setY: yInteger`

What You Should Know

- A class is defined by sending a message to its superclass
- Classes are defined inside packages
- Methods are public
- By default a method returns the receiver, `self`
- Class methods are just methods of the class side



Resources

- Pharo Mooc - W1S06 Videos
- Pharo by Example <http://books.pharo.org>



A course by Stéphane Ducasse
<http://stephane.ducasse.free.fr>

Reusing some parts of the Pharo Mocc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse
<http://mocc.pharo.org>



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>