



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

First Look at Class and Method Definitions

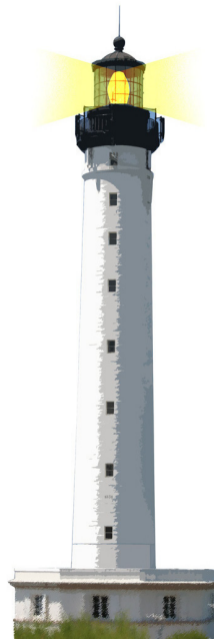
Stéphane Ducasse

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



<http://www.pharo.org>

W1S10

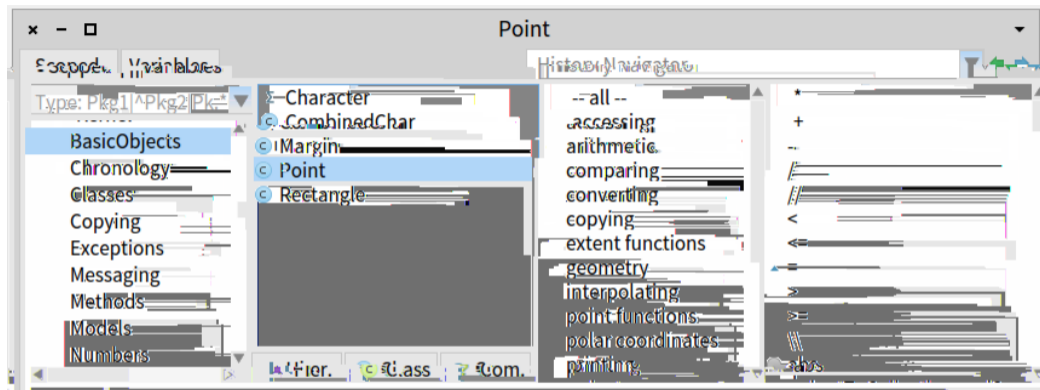


Class and Method Definitions in Pharo

- 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 with the class browser open to the `Point` class. The class browser is titled "Point" and shows a hierarchy of classes: `Character`, `CombinedChar`, `Margin`, `Point`, and `Rectangle`. The `Point` class is selected. The right pane shows the class definition, which is currently empty. The bottom pane shows the class definition code:

```
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
- Class Browser:**
 - ExactFloatPrintPolicy
 - FloatPrintPolicy
 - Integer (selected and highlighted in pink)
 - Magnitude
 - Number
 - Float
 - Fraction
 - ScaledDecimal
- History Navigator:**
 - accessing
 - arithmetic
 - bit manipulation
 - comparing
 - converting
 - converting-arrays
 - enumerating
 - filter streaming
 - mathematical func (highlighted in pink)
- Method List:**
 - factorial (selected)
 - gcd:
 - nthRoot:
 - nthRootRounded:
 - nthRootTruncated:
 - raisedTo:modulo:
 - raisedToInteger:modulo:
 - sqrt
 - take:
- 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

Integer>>#factorial

Scoped Variables History Navigator

Type: Pkg1|^Pkg2|P|

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

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

Hier. Class Com.

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

factorial

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

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 with the Point class editor open. The class name is 'Point class>>#x,y:'. The class browser on the left shows the hierarchy: Pkg1 | ^Pkg2 | jobs | JobsTests | Kernel | BasicObjects | Chronology | Classes. The class list in the center shows: Character, CombinedChar, Margin, Point, Rectangle. The history navigator at the top shows: -- all --, instance creation, *System-Setting. The code editor at the bottom shows the following code:

```
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://mooc.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/>