

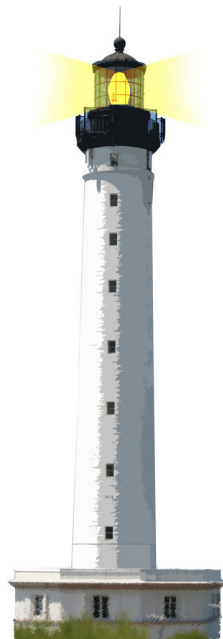
# Class and Method Definitions

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W1S06



<http://www.pharo.org>



# Class and Method Definitions in Pharo

- classes and methods are defined within tools
- there is no dedicated syntax



# Class Definition Template

The screenshot shows a software development environment window titled "BasicObjects". The interface includes a "Scoped" tab, a "Variables" tab, and a "History Navigator" search bar. A tree view on the left lists various categories: BasicObjects, Chronology, Classes, Copying, Exceptions, Messaging, Methods, Models, and Numbers. The "BasicObjects" category is selected, and a list of subclasses is displayed: Character, CombinedChar, Margin, Point, and Rectangle. Below the tree view, there are buttons for "Hier.", "Class", and "Com.". The main area of the window displays a class definition template for an "Object subclass".

```
Object subclass: #NameOfSubclass
  instanceVariableNames: ''
  classVariableNames: ''
  category: 'Kernel-BasicObjects'
```

At the bottom of the window, there is a status bar showing "1/4 [1]" and a checkbox for "Format as you read". A warning message "Excessive number of methods" is displayed with a red 'X' icon. To the right of the warning, there is a "Helpful?" label with green and red thumbs-up/down icons.

# Class Definition in Pharo

The screenshot shows the Pharo IDE interface. At the top, a window titled "Point" is open. On the left, a "Scopes" pane shows a tree of packages, with "BasicObjects" selected. Below it, a list of classes includes "Character", "CombinedChar", "Margin", "Point" (highlighted), and "Rectangle". The "History Navigator" on the right lists various methods like "accessing", "arithmetic", "comparing", etc. The main editor area displays the class definition for `#Point`:

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

At the bottom, a status bar shows "1/4 [1]" and a warning icon with the text "Excessive number of methods".

# 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

- Methods are public
- Methods are virtual (i.e., looked up at runtime)
- By default return `self`

```
messageSelectorAndArgumentNames  
  "comment stating purpose of message"
```

```
| temporary variable names |  
statements
```



# Method Definition in Pharo

The screenshot shows the Pharo IDE interface with the title bar "Integer>>#factorial". The interface is divided into several panes:

- Left Pane:** Contains a "Type: Pkg1|^Pkg2|P" dropdown and a tree view of the package structure. The "Numbers" package is selected, showing sub-packages like "Kernel-Tests", "Keymapping-Cor", and "Keymapping-Key".
- Class Browser:** A list of classes and protocols. "Integer" is selected under the "mathematical func" protocol.
- History Navigator:** A list of methods. "factorial" is selected.
- Method Definition:** The main area shows the source code for the `factorial` method:

```
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'
```
- Bottom Bar:** Shows the cursor position "1/6 [1]" and a checkbox for "Format as you read" which is currently unchecked.

# 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

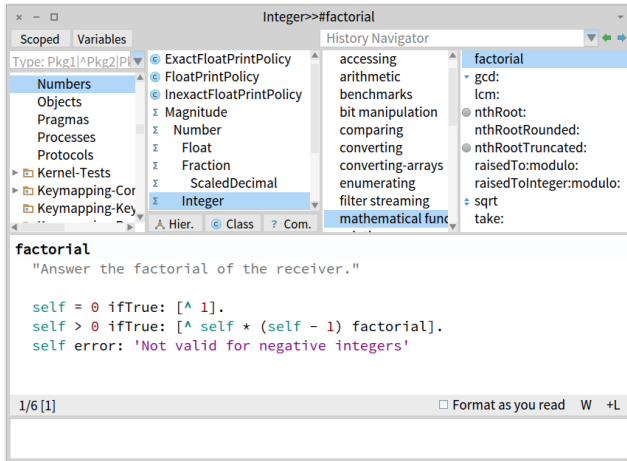
In this lecture, a method will be displayed as

```
Integer >> factorial
"Answer the factorial of the receiver."
self = 0 ifTrue: [ ^ 1 ].
self > 0 ifTrue: [ ^ 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



In Pharo, the method belongs to the selected class

# 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

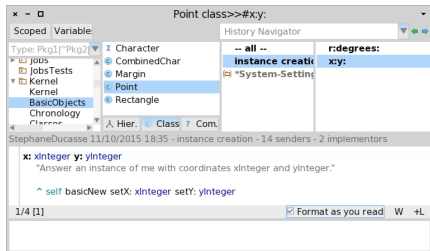
```
Game >> initializePlayers  
self players  
at: 'tileAction'  
put: ( MITileAction director: self )
```

is equivalent to

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



# Class Methods



- 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



A course by



and



in collaboration with



Inria 2020

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/>