Pharo 9 by Example

Jose M. Cano V.

Universidad Nacional de San Agustin de Arequipa
Facultad de Produccion y Servicios
Escuela Profesional de Ingenieria de Sistemas

Noviembre 12 2022

Outline

Chapter 1

About the book

- Cover of the new system browser, Calypso, as well as the Pharo Launcher
- Simplified the SUnit chapter
- improvements as result of many people's work

What is Pharo?

- Pharo is a pure object-oriented programming language and powerful environment, it is an IDE and OS together.
- A dynamically-typed language
- You can modify their code as it executes.
- It is a pure, dynamic, and simple object-oriented language, consisting entirely of objects.

Chapter 1

Try not to care. Beginning Pharo programmers often have trouble because they think they need to understand all the details of how a thing works before they can use it.

The Pharo community

- The Pharo community is friendly and active.
- http://www.github.com/pharo-project/pharo
- Pharo has an active on Discord server

Outline

Chapter 2

Installing Pharo

- Installing Pharo: Pharo does not need to install anything in your system, as it's perfectly capable of running standalone.
- Pharo Launcher: Its a cross-platform application that facilitates the management of multiple Pharo images and VMs.
- Zeroconf scripts: Collection of scripts to download specific versions of Pharo. To download the latest Pharo 9.0 full system, use the following snippet

  : $wget - O - get.pharo.org/90 + vm|bash$

  Then you can execute the following to start the image:

  $./pharo - uiPharo.image$

  At present these scripts will only work on a a macOS or Linux system.

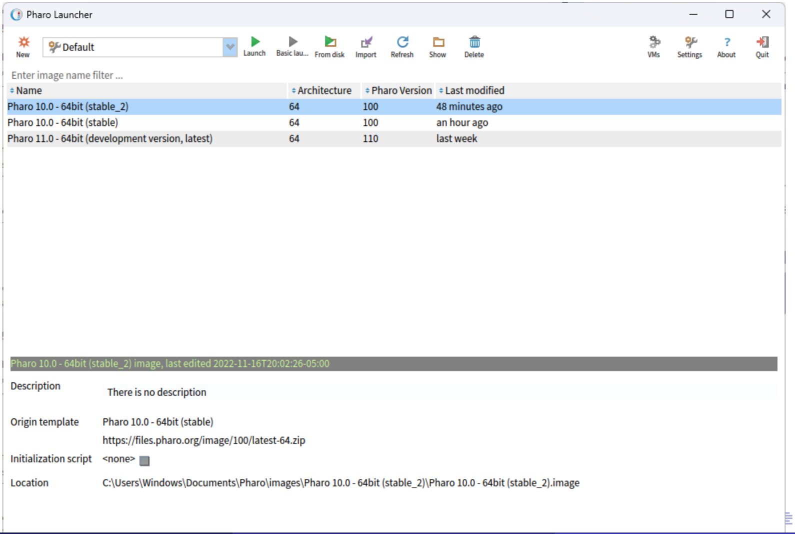
Chapter 2

File components

- Virtual Machine The VM is the execution engine (similar to Java VirtualMachine).
- The Image file On one operating system it can be used on any other operating system with a supported virtual machine. An image is basically a container for virtual objects.
- The change file Records all source code modifications The change file is always associated with The Image file
- The sources file This file is important because the image file format stores only objects, including compiled methods and their bytecode, and not their source code.

Chapter 2: Pharo Launcher

Pharo Launcher is a tool that helps you download and manage Pharo images. It is very useful for getting new versions of Pharo.



Name	Architecture	Pharo Version	Last modified
Pharo 10.0 - 64bit (stable_2)	64	100	48 minutes ago
Pharo 10.0 - 64bit (stable)	64	100	an hour ago
Pharo 11.0 - 64bit (development version, latest)	64	110	last week

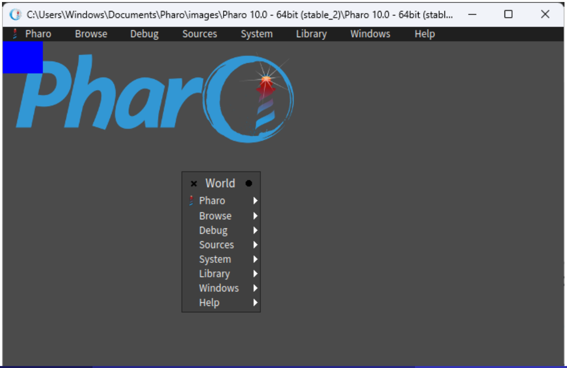
Pharo 10.0 - 64bit (stable_2) image, last edited 2022-11-16T20:02:26-05:00

Description	There is no description
Origin template	Pharo 10.0 - 64bit (stable)
	https://files.pharo.org/image/100/latest-64.zip
Initialization script	<none>
Location	C:\Users\Windows\Documents\Pharo\images\Pharo 10.0 - 64bit (stable_2)\Pharo 10.0 - 64bit (stable_2).image

Outline

World Menu

- Once Pharo is running, you should see a single large window, possibly ontaining some open playground windows. Clicking anywhere on the background of the Pharo window will display thev World Menu, which contains many of the Pharo tools, utilities, and settings. The world menu provides you with a simple way to access many of the tools that Pharo offers.

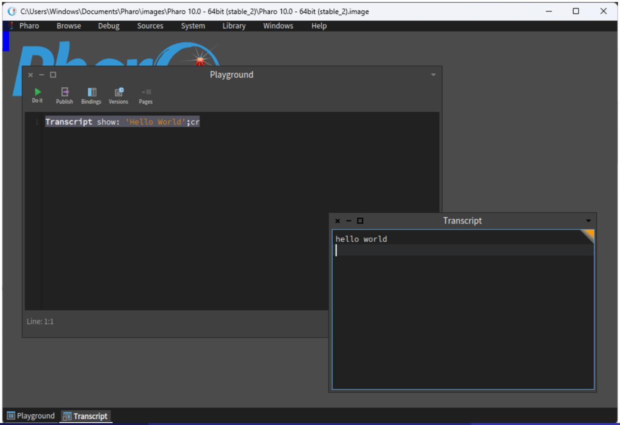


Chapter 3

Interacting with Pharo:
Pharo is designed to work with a three-button mouse to click, action-click, or meta-click. If you don't have a three button mouse, you can use the modifier keys to get the same effect.
click (or left-click)
action-click (or right-click)
meta-click: meta-click on any object displayed in the image to activate the "Morphic halo"

Playgrounds and Transcripts

- The transcript is an object that is often used to issue system messages. It is a type of "system console"
- Workspaces are useful for writing portions of Smalltalk code that you would like to experiment with.


C:\Users\Windows\Documents\Pharo\images\Pharo 10.0 - 64bit (stable_2)\Pharo 10.0 - 64bit (stable_2).image
Pharo Browse Debug Sources System Library Windows Help
Playground
Do it Publish Bindings Versions Pages
Line: 1:1
Transcript
Playground Transcript

```
Transcript show: 'Hello World';cr
```

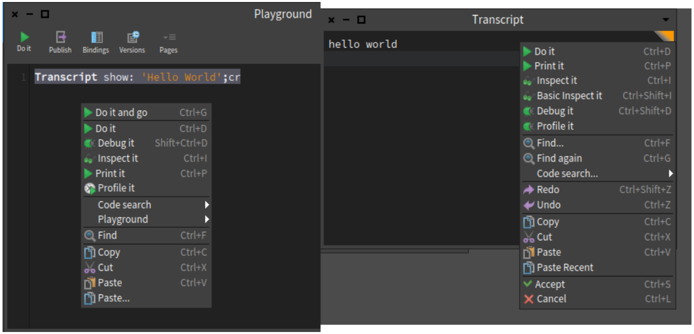
```
hello world
```

Keyboard shortcuts

Contains over 200 shortcuts

143 000 methods contained in the Pharo image. **Doing vs. printing**

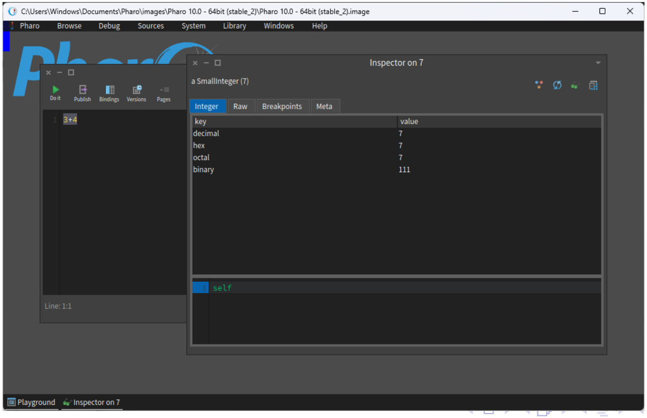
Type the expression 3 + 4 into the playground. Now Do it with the keyboard shortcut Cmd-D.


```
Transcript show: 'Hello World';cr
```

```
hello world
```

Inspect

The inspector is an extremely useful tool that allows you to browse and interact with any object in the system. The inspector features specific tabs that display different information and views about an object depending on the type of object you are inspecting.



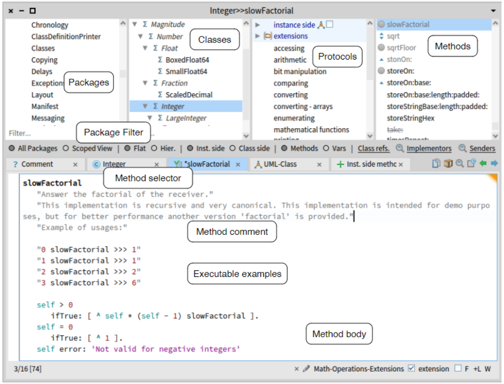
```
3+4
```

key	value
decimal	7
hex	7
octal	7
binary	111

```
self
```

Calypso: the System Browser

The System Browser, also known as the "Class Browser", is one of the key tools used for programming in Pharo.



```
slowFactorial
   "Answer the factorial of the receiver."
   "This implementation is recursive and very canonical. This implementation is intended for demo purpo
ses, but for better performance another version 'factorial' is provided."
   "Example of usages:"

   "0 slowFactorial >>> 1"
   "1 slowFactorial >>> 1"
   "2 slowFactorial >>> 2"
   "3 slowFactorial >>> 6"

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

Calypso: the System Browser

- The first panel lists all known packages, which contain groups of related classes.
- The second panel displays a list of all the classes in the selected package.
- The third panel shows the protocols of the currently selected class. These are convenient groupings of related methods.
- The fourth panel is the methods.

Other operations

Other right-click options that may be used are the following:

- Do it and go additionally opens a navigable inspector on the side of the playground. It allows us to navigate the object structure.
- Basic Inspect it opens the classic inspector that offers a more minimal interface and live updates of changes to the object.
- Debug it opens the debugger on the code.
- Profile it profiles the code with the Pharo profile tool, showing you how much time is spent for each message that is sent.

Outline

Chapter 4

Navigating using the System Browser

- Open the Browser
- Filter packages
- Expand the Kernel package and select the Objects element
- Select the Object class
- Select the printing protocol
- Select the printString method

Chapter 4

Finding classes

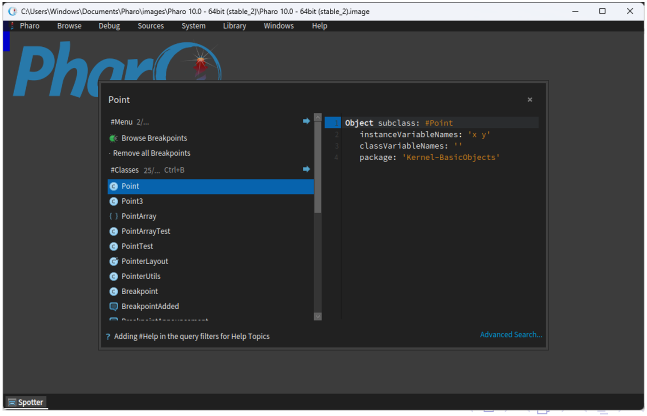
A second way is to send the browse message to an instance or the class itself, asking it to open a Browser on itself. Suppose we want to browse the class Point:

- Using the Message browse: Type Point browse into a Playground and then Do it. A Browser will open on the Point class.
- Using Cmd-B: There is also a keyboard shortcut Cmd-B that you can use in any text pane; select the word and press Cmd-B.

Using Spotter

The method finder will display a list of all method names that contain the search word.

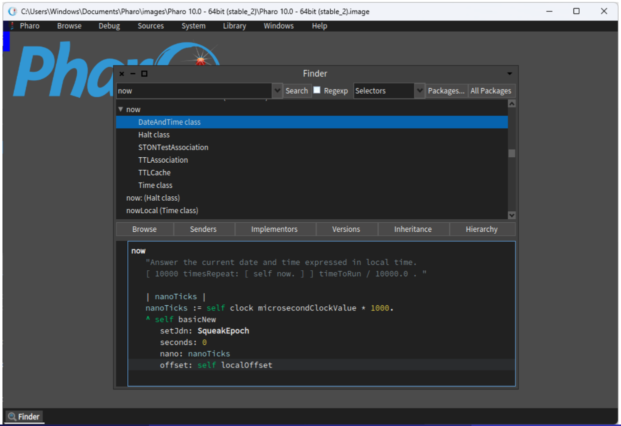
The fastest and best way to find a class is to use Spotter. Pressing Shift-Enter opens Spotter [1]


C:\Users\Windows\Documents\Pharo\images\Pharo 10.0 - 64bit (stable_2)\Pharo 10.0 - 64bit (stable_2).image
Pharo Browse Debug Sources System Library Windows Help
Point
#Menu 2/...
Browse Breakpoints
Remove all Breakpoints
#Classes 25/... Ctrl+B
Point
Point3
PointArray
PointArrayTest
PointTest
PointerLayout
PointerUtils
Breakpoint
BreakpointAdded
? Adding #Help in the query filters for Help Topics
Advanced Search...
Spotter

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

Using 'Find class' in the System Browser

In the System Browser you can also search for a class by its name.
In the System Browser, click anywhere in the package pane or the class pane, and launch the Class Search window by typing Cmd-F, or selecting Find class from the right-click context menu.



```
▼ now
    DateAndTime class
    Halt class
    STONTestAssociation
    TTLAssociation
    TTLCache
    Time class
  now: (Halt class)
  nowLocal (Time class)
```

```
now
	"Answer the current date and time expressed in local time.
	[ 10000 timesRepeat: [ self now. ] ] timeToRun / 10000.0 . "

	| nanoTicks |
	nanoTicks := self clock microsecondClockValue * 1000.
	^ self basicNew
		setJdn: SqueakEpoch
		seconds: 0
		nano: nanoTicks
		offset: self localOffset
```

Finding methods

Spotter

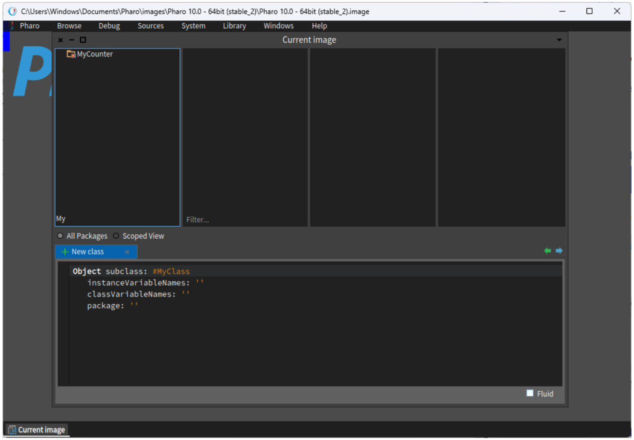
- The Spotter is a powerful tool to navigate the system and find information.

Finder

- The Finder allows you to search for classes, methods, and more. Also allows you to find methods based on the object receiving the message, the arguments of the message, and the returned object.

Outline

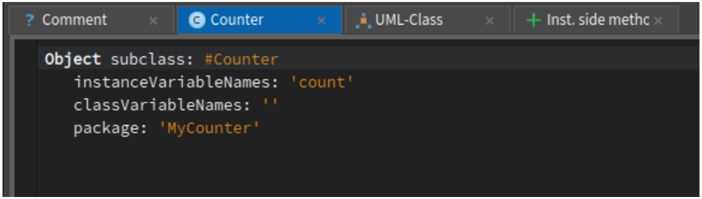
Create a package and class



```
Object subclass: #MyClass
    instanceVariableNames: ''
    classVariableNames: ''
    package: ''
```

Create a package and class

As we are disciplined developers, we will add a comment to our Counter class by clicking the Comment pane and the Toggle Edit / View comment toggle.



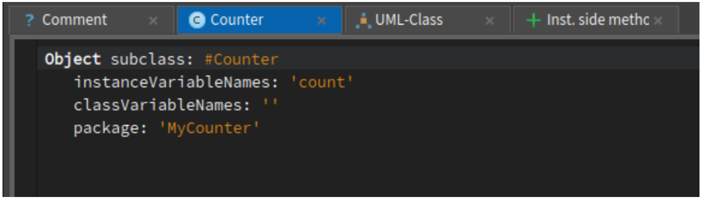
```
Object subclass: #Counter
    instanceVariableNames: 'count'
    classVariableNames: ''
    package: 'MyCounter'
```

Create a package and class

You should get the following class definition:

```
Object subclass: #Counter
       instanceVariableNames: 'count'
       classVariableNames: ''
       package: 'MyCounter'
```

- We now have a class definition for the Counter class. To define it in our system we still have to compile it.
- Once the Counter class is compiled, it will be added to the system.



```
Object subclass: #Counter
    instanceVariableNames: 'count'
    classVariableNames: ''
    package: 'MyCounter'
```

Chapter 5

Defining protocols and methods

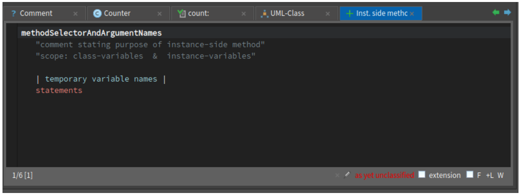
The method editor selected and ready to define a method.

- Everything is an object
- Instance variables are completely private to the object
- The only way to interact with an object is by sending messages to it

Create a method

The method editor selected and ready to define a method.

- Everything is an object
- Instance variables are completely private to the object
- The only way to interact with an object is by sending messages to it

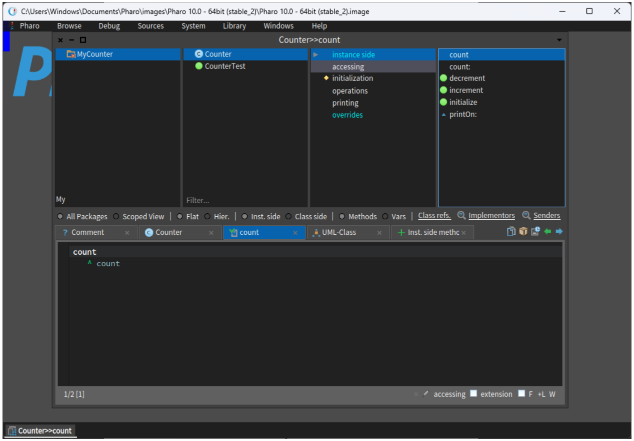


```
methodSelectorAndArgumentNames
    "comment stating purpose of instance-side method"
    "scope: class-variables  &  instance-variables"

    | temporary variable names |
    statements
```

Adding a setter method

Complementing the getter method we find the setter method. These are used to change the value of an instance variable from outside the object.


C:\Users\Windows\Documents\Pharo\images\Pharo 10.0 - 64bit (stable_2)\Pharo 10.0 - 64bit (stable_2).image
Pharo Browse Debug Sources System Library Windows Help
Counter>>count
MyCounter
Counter
CounterTest
instance side
accessing
initialization
operations
printing
overrides
count
count:
decrement
increment
initialize
printOn:
My
Filter...
All Packages Scoped View Flat Hier. Inst. side Class side Methods Vars Class refs. Implementors Senders
Comment Counter count UML-Class Inst. side methc
1/2 [1] accessing extension F +L W
Counter>>count

```
count
	^ count
```

Chapter 5

Define Class Test case

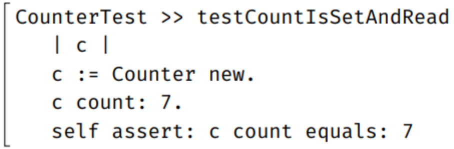
Our test cases, written as methods, need to live inside a test class that inherits by TestCase. So we define a class called CounterTest as follows:

```
TestCase subclass: #CounterTest
      instanceVariableNames: ''
      classVariableNames: ''
      package: 'MyCounter'
```

Chapter 5

Define the following method for our test case

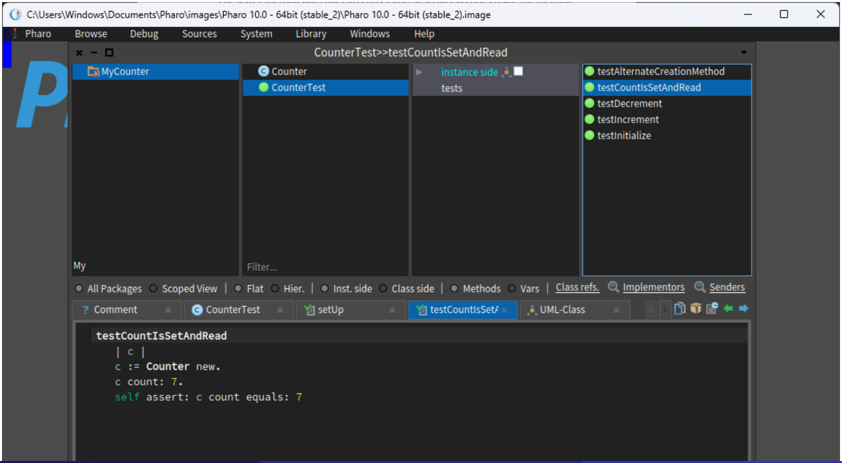
The assert:equals: message is a message implemented in our test class. check a fact (in this case, that two objects are equal), and the test will fail if the fact it isn't true



```
CounterTest >> testCountIsSetAndRead
   | c |
   c := Counter new.
   c count: 7.
   self assert: c count equals: 7
```

Chapter 5

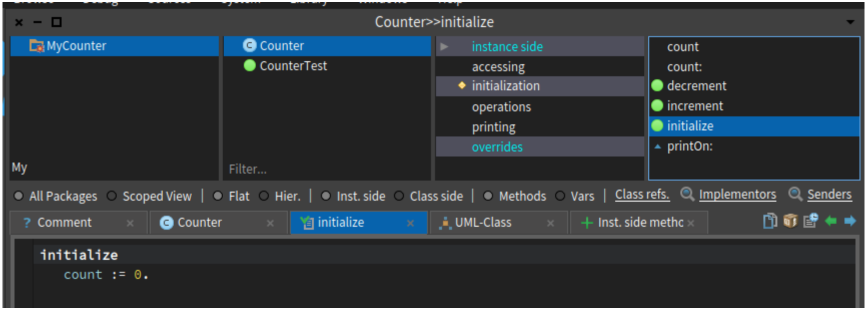
To maximize their potential, unit tests should be fast, repeatable, independent of any human interaction, and cover a single functionality. Tests for a class named MyClass belong to a class classified as MyClassTest, which must be presented as a subclass of TestCase.


C:\Users\Windows\Documents\Pharo\images\Pharo 10.0 - 64bit (stable_2)\Pharo 10.0 - 64bit (stable_2).image
Pharo Browse Debug Sources System Library Windows Help
CounterTest>>testCountIsSetAndRead
MyCounter
Counter
CounterTest
instance side
tests
testAlternateCreationMethod
testCountIsSetAndRead
testDecrement
testIncrement
testInitialize
My
Filter...
All Packages | Scoped View | Flat | Hier. | Inst. side | Class side | Methods | Vars | Class refs. | Implementors | Senders
Comment | CounterTest | setUp | testCountIsSetA | UML-Class

```
testCountIsSetAndRead
    | c |
    c := Counter new.
    c count: 7.
    self assert: c count equals: 7
```

Instance initialization method

The initialization message is sent to the newly created instance so it must be defined on the instance side like any other method. The initialization method is responsible for setting the default values of the instance variables.



```
initialize
    count := 0.
```

Conclusion

- In this tutorial, you learned how to define packages, classes, methods, and tests.
- Everything is an object. Primitive entities like integers are objects, but classes are also first-order objects.
- Everything happens by sending messages We don't "call methods", "we send messages". The recipient then chooses his own method of replying to the message.

References

Stéphane Ducasse, Gordana Rakic, Sebastijan Kaplar, and Quentin Ducasse.
Pharo 9 by example.
BoD-Books on Demand, 2022.