



The Pharo Module System

Guille Polito

ESUG'26 - Plovdiv, Bulgaria

guillermo.polito@inria.fr
@guillep



Quick About Me: Guille

guillermo.polito@inria.fr
@guillep



- Pronounced *gife* (guichet in FR, ~ghisheh in EN?)
- **Now:** Researcher at Inria - Lille
- Pharo Contributor since ~2010
- **Keywords:** compilers, testing, test generation
- **Interests:** tooling, benchmarking, 日本語, board games, batman, concurrency, chess



If any of that interests you, come talk to me!



“Can we have namespaces?”

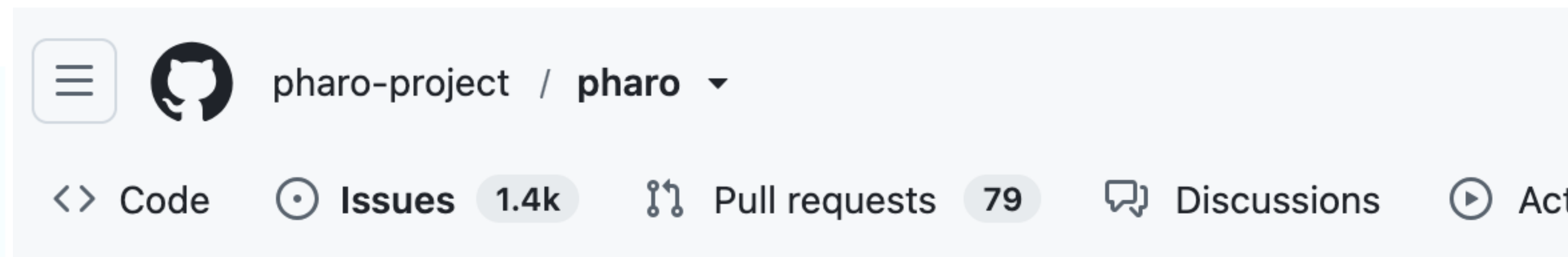
Discussion:

[Pharo-project] GSoC Project for Namespaces

James Foster

16 years ago

Germán Leiva was accepted in Google Summer of Code 2010 and developed a Namespaces implementation for Pharo Smalltalk. A video at



Can we have namespaces? #1356

Closed



privat opened on Apr 26, 2023

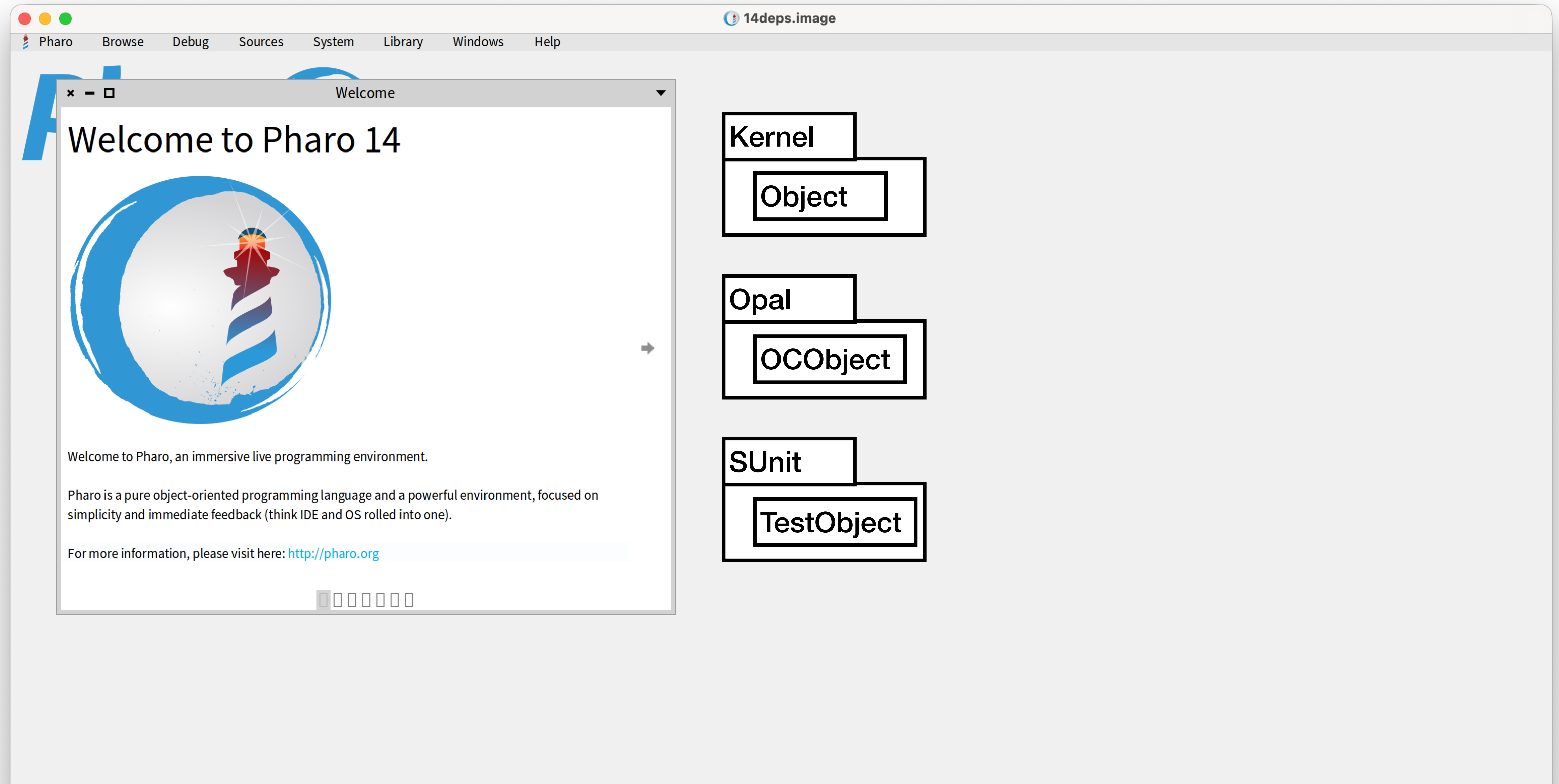
Namespaces are a classical approach to deal with names conflicts caused by the *two-uppercase-prefix* convention on classnames works locally.

No

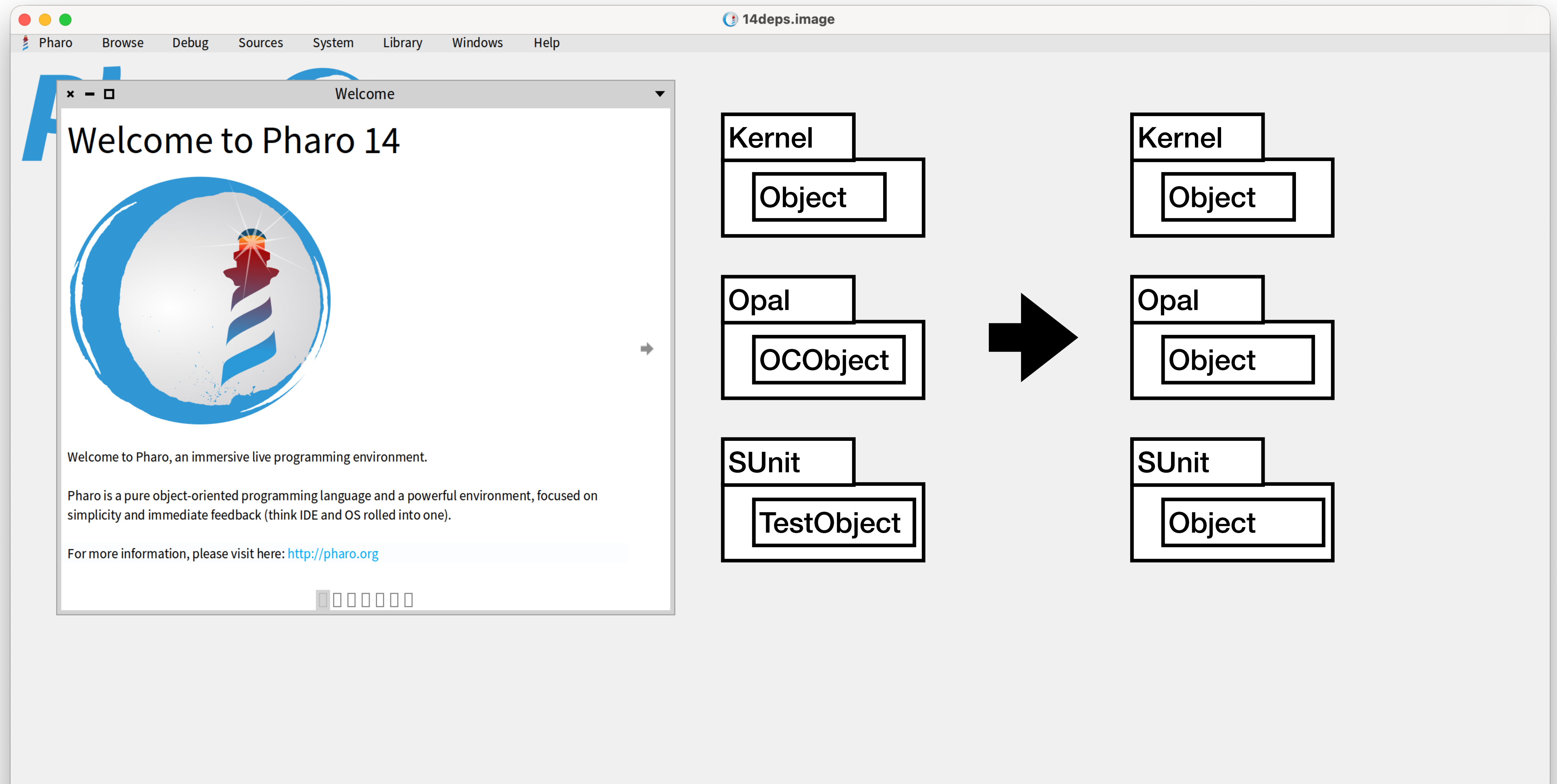
~~No~~

Namespaces are not enough

Namespaces = Name Spaces



Namespaces closeTo: avoidPrefixes



Porting old code

Pharo

Browse

Debug

Sources

System

Library

Windows

Help

14deps.image

Welcome

Welcome to Pharo 14



Welcome to Pharo, an immersive live programming environment.

Pharo is a pure object-oriented programming language and a powerful environment, focused on simplicity and immediate feedback (think IDE and OS rolled into one).

For more information, please visit here: <http://pharo.org>

Kernel

Opal

SUnit

Porting old code

14deps.image

Pharo Browse Debug Sources System Library Windows Help

Welcome

Welcome to Pharo 14



Welcome to Pharo, an immersive live programming environment.

Pharo is a pure object-oriented programming language and a powerful simplicity and immediate feedback (think IDE and OS rolled into one).

For more information, please visit here: <http://pharo.org>

Kernel

SqueakSource

www.squeaksource.com/@sMKYqHczorB2b0hc/oHwVMdqc

Tech Research Admin Japanese Cosas Argentinas Stages vm Pharo GH evref-vm druid mutalk work Create a conference COMPILER Course OBS-gp Tab

SqueakSource

Home Projects Tags Members Groups Help

Actions

[RSS feed](#)

[Register Member](#)

[Register Group](#)

[Register Project](#)

Authentication

[Login](#)

Home

Welcome to SqueakSource, the central repository for Squeak and Pharo. To get started, register your personal account. You'll get all the necessary permissions to create your account, projects and versions. Detailed instructions can be found on the Help page.

This service is provided at no cost to maintain suitable private back-up and ensure reliable operation, but accessibility is not guaranteed and users are encouraged to set up their own SqueakSource server. It is self-hosted and can be downloaded from the SqueakMap.

Please report any problems or suggestions to the [Squeak mailing list](#) or the [Squeak box administration mailing list](#). Enjoy!

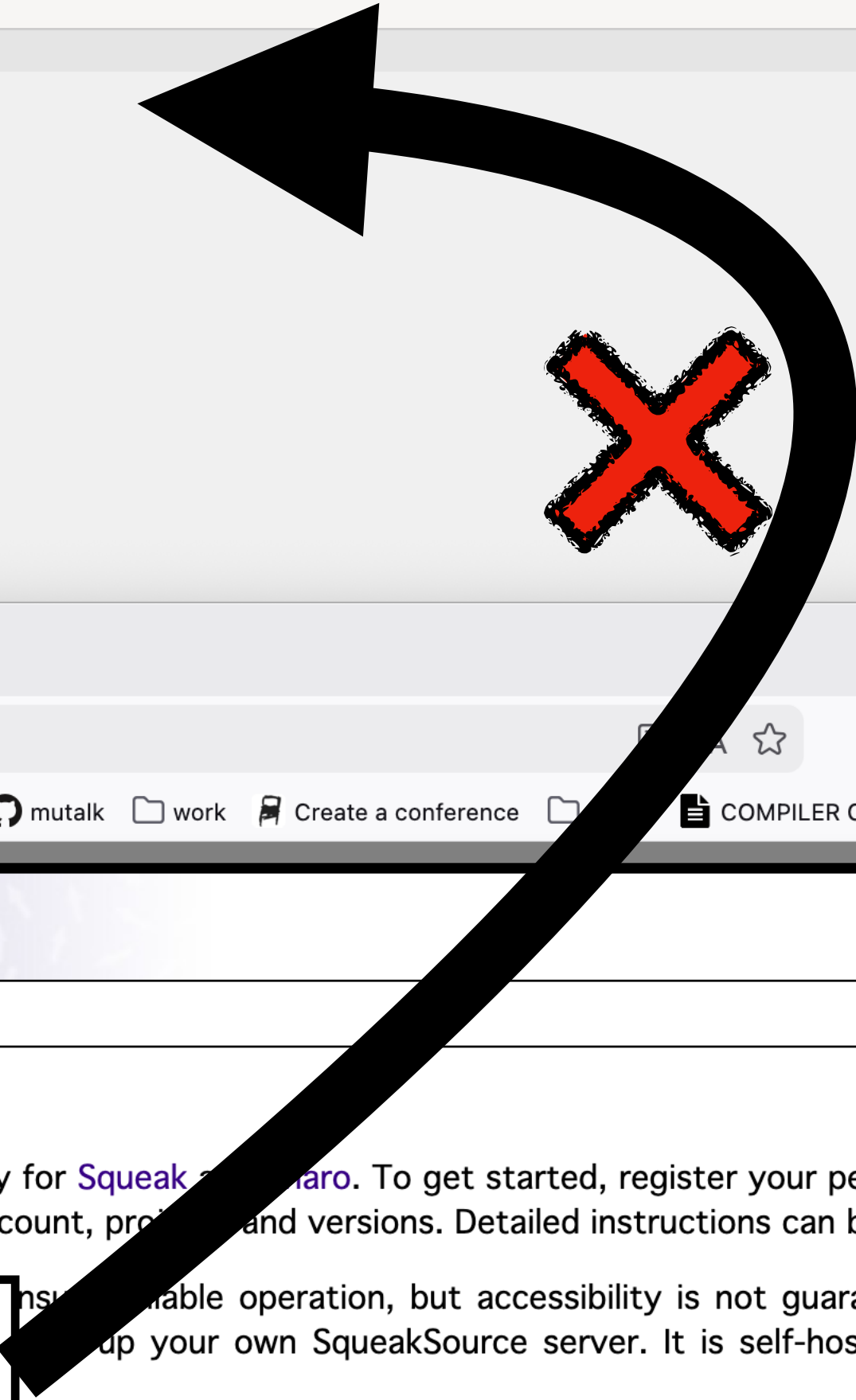
We gratefully acknowledge the [Software Composition Group](#) and the [University of Bern](#) for hosting and supporting this service for many years. Thank you to the [Squeak community](#) for its ongoing support.

Statistics

Members (4513)

Recently Joined: [ciuca](#), [Vladimir](#), [Michael Wilson](#), [Michael Wilson](#), ...

MyOldLib



Me @ the Future



Modularity by Construction

Guille Polito

ESUG'26 - Plovdiv, Bulgaria

guillermo.polito@inria.fr

@guillep

Inria

Evref

cnrs

CRISTAL

Université de Lille

2026-ESUG-Modularity

Présentation Zoom Ajouter une diapositive Lire Tableau Graphique Texte Figure Données multimédias Commentaire Partager

57 %

+

1

2

3

4

5

6

7

8

9

10

11

12

Titre de la présentation

Aspect

☒ Titre

☒ Corps

☒ Numér

Arrière-pl

Star

Remplissa

Rempliss

Modifi

Me @ the Future Talking about Modularity



2026-ESUG-Modularity

Présentation Zoom Ajouter une diapositive Lire Tableau Graphique Texte Figure Données multimédias Commentaire Partager

1 Modularity by Construction

2 Quick About Me: Guille

3

4

5

6

7

8

9

10

11

12

Bootstrap

Modularity by Construction

Refactoring

Code Migration

Compatibility

ESUG'26 - Plovdiv, Bulgaria

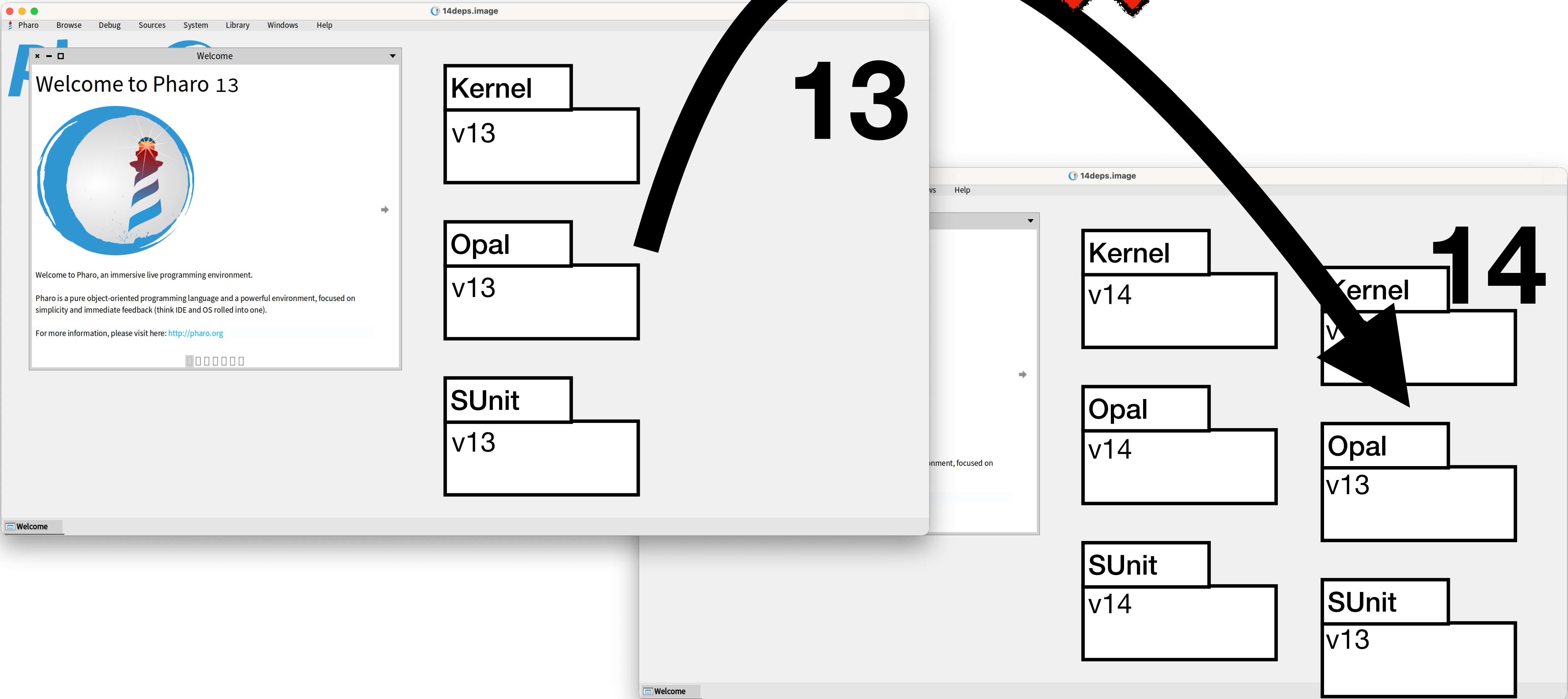
Inria Evref cnrs

CRISTAL Centre de Recherche en Informatique, Signal et Automatique de Lille Université de Lille

1

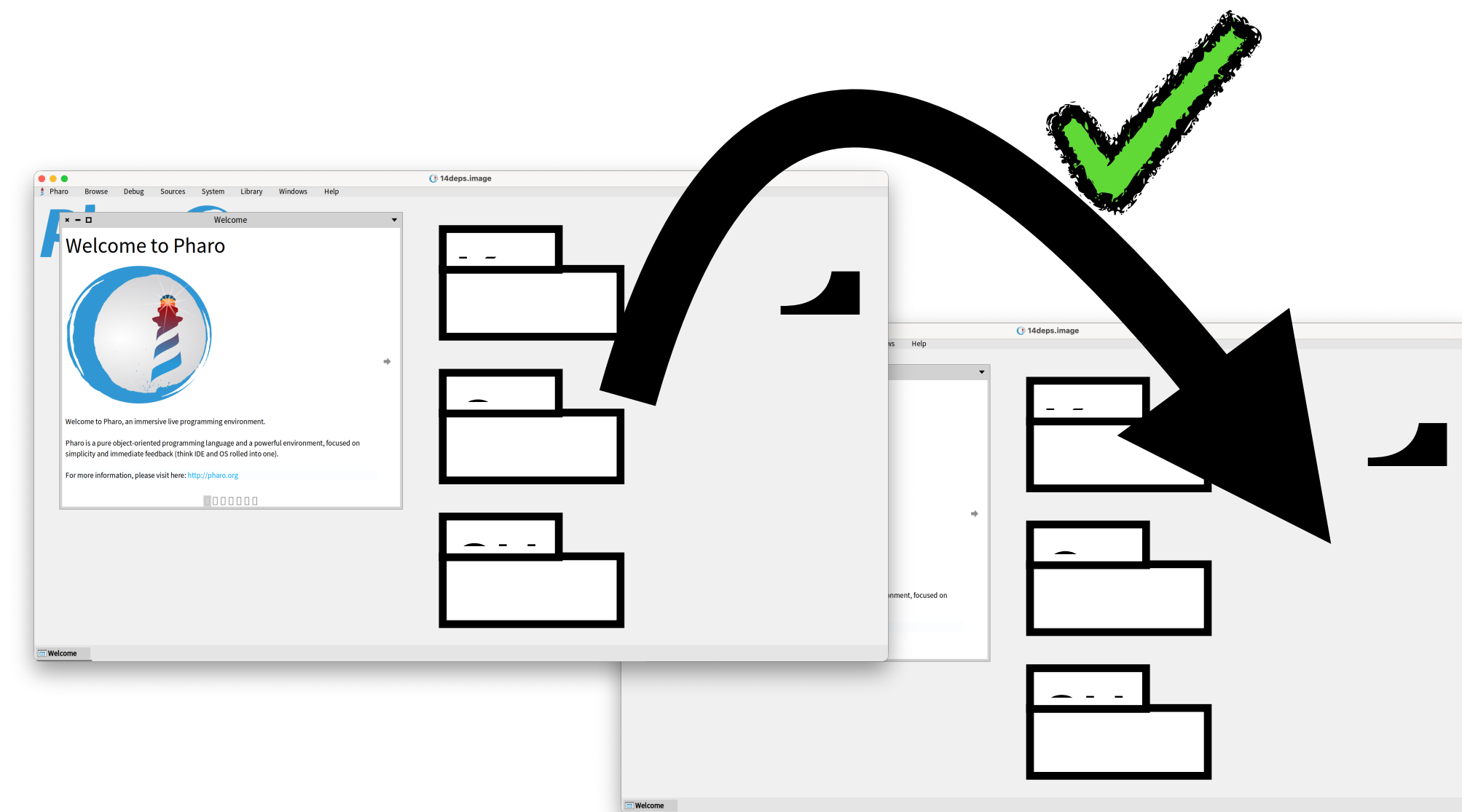
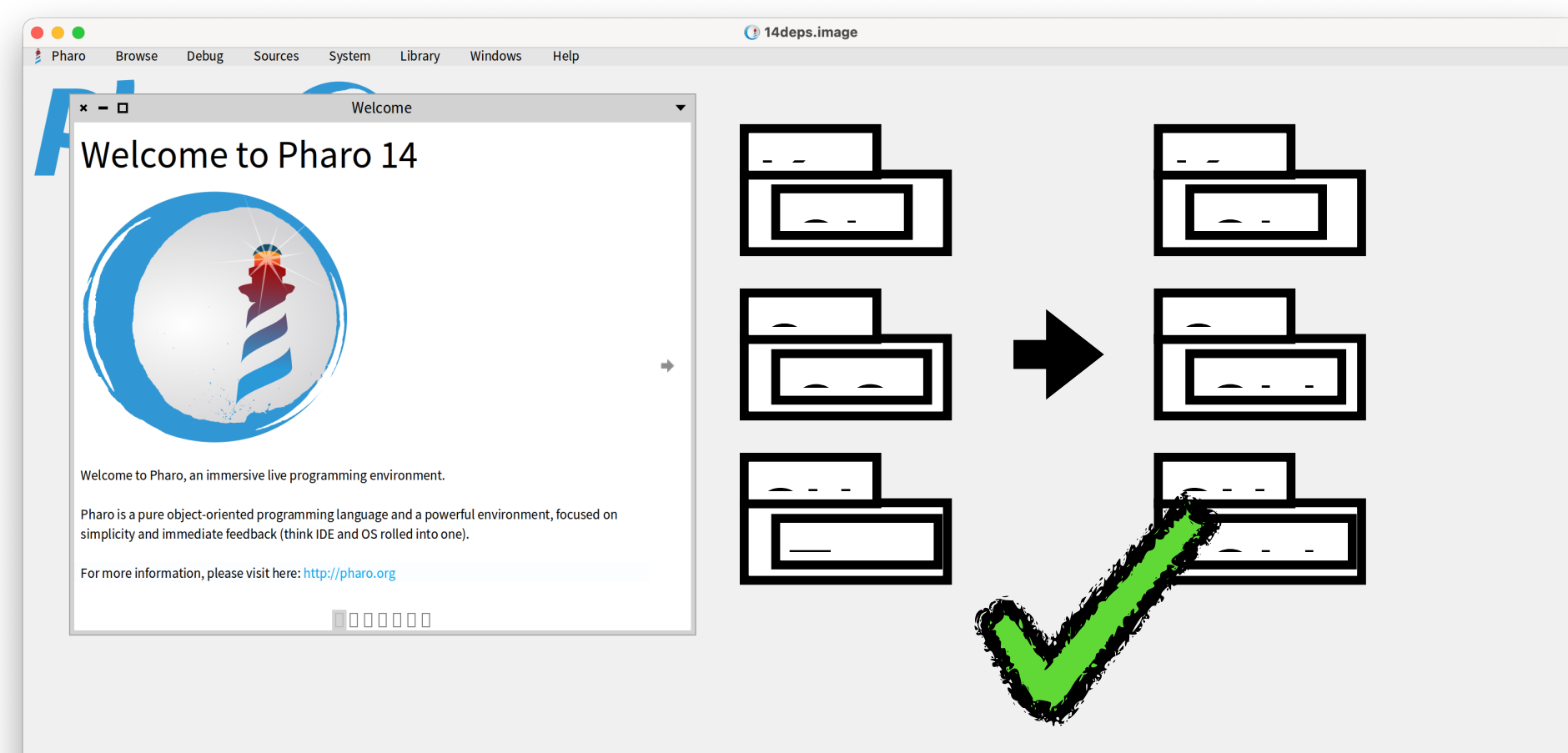
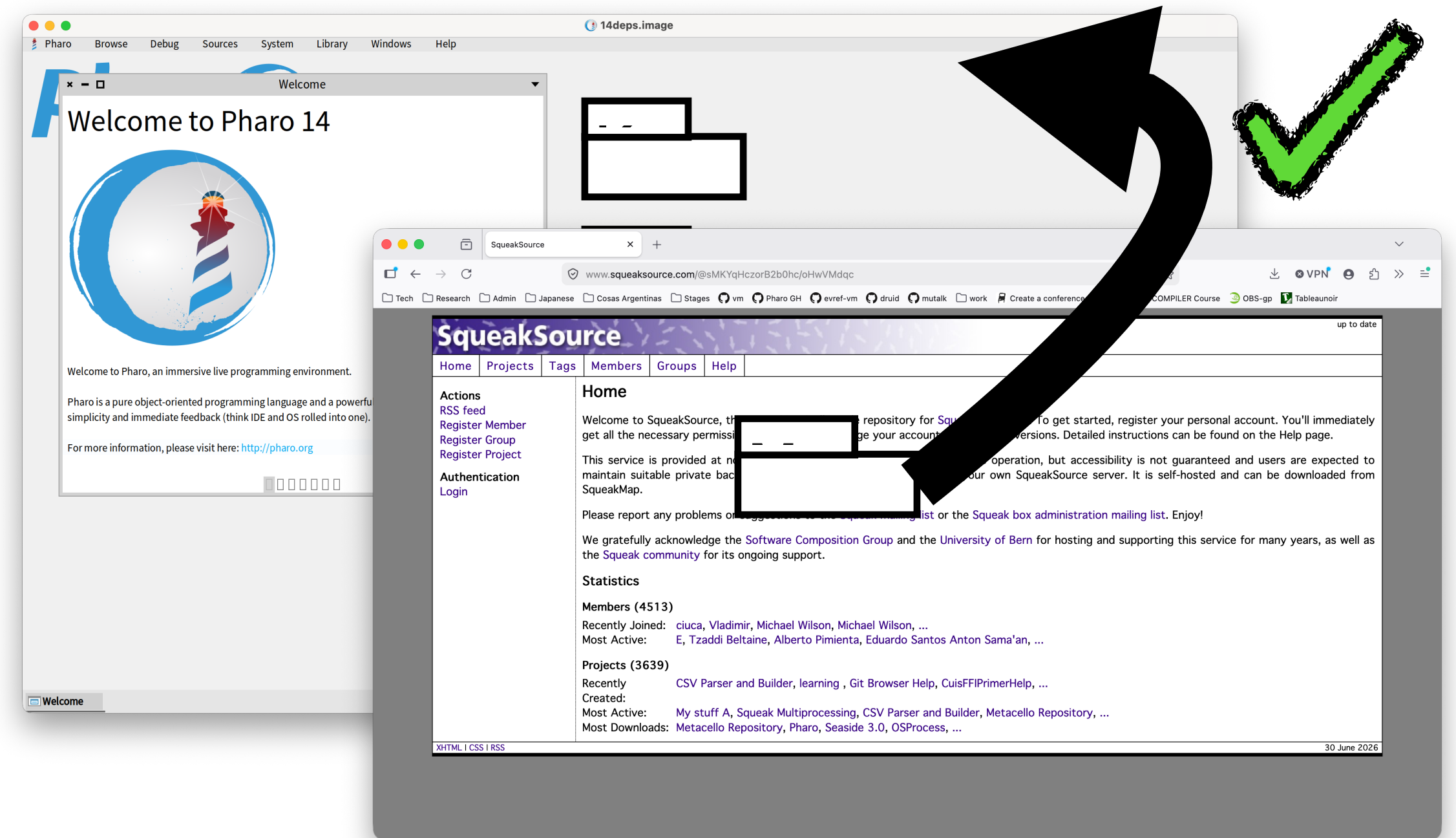
Modifi

Side by Side Versions



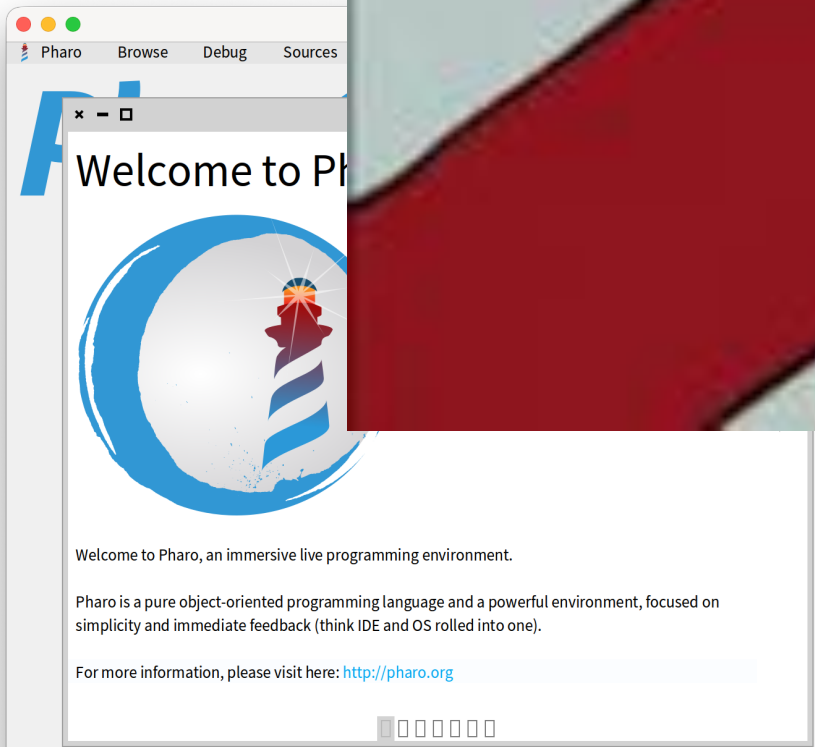
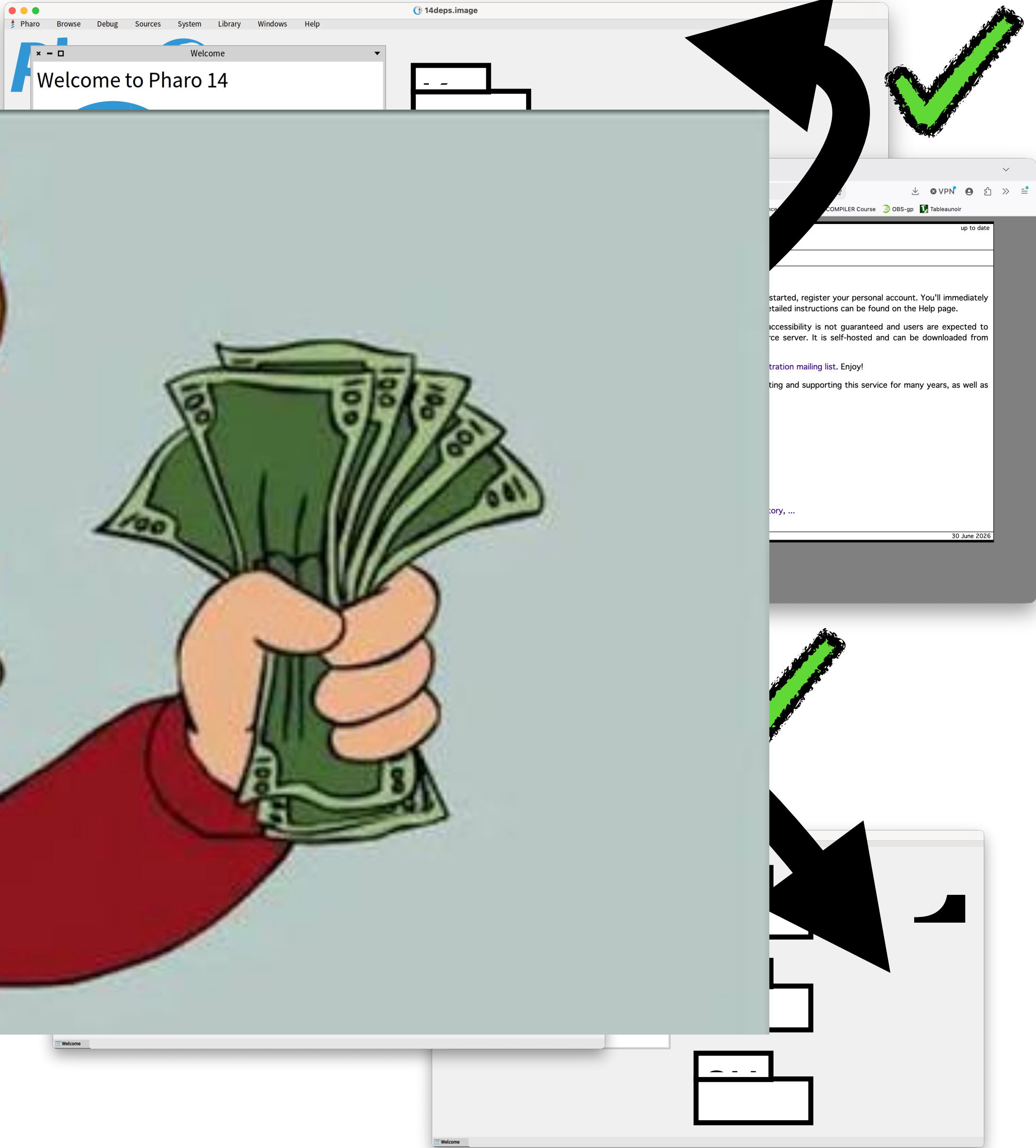
Goal

- Name spaces, but **not only**
- Ease of **code migration and porting**
- Simplify **compatibility**
- As much **backward** compatible as possible

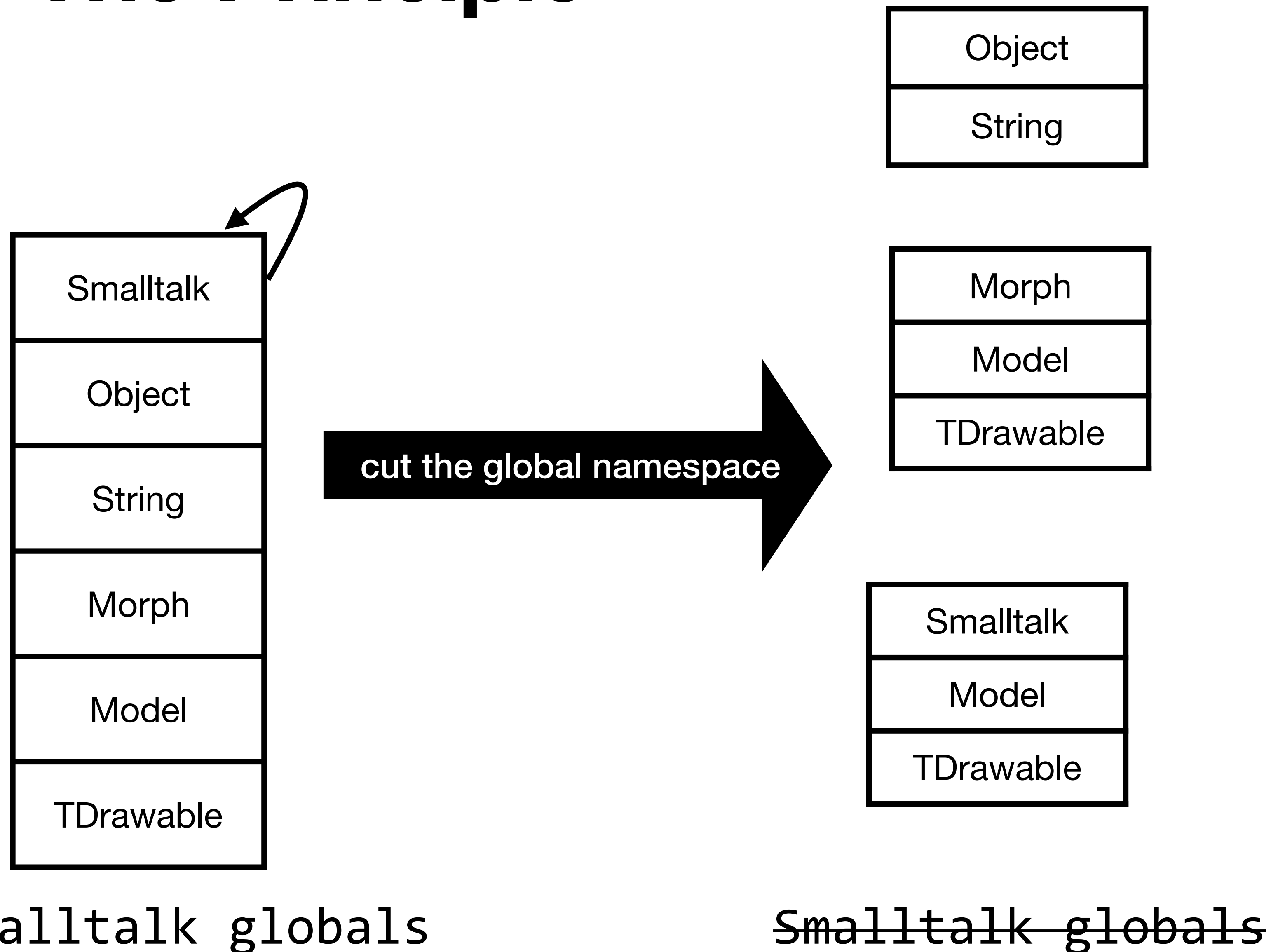


Goal

- Name s
- Ease of
- Simplify
- As muc

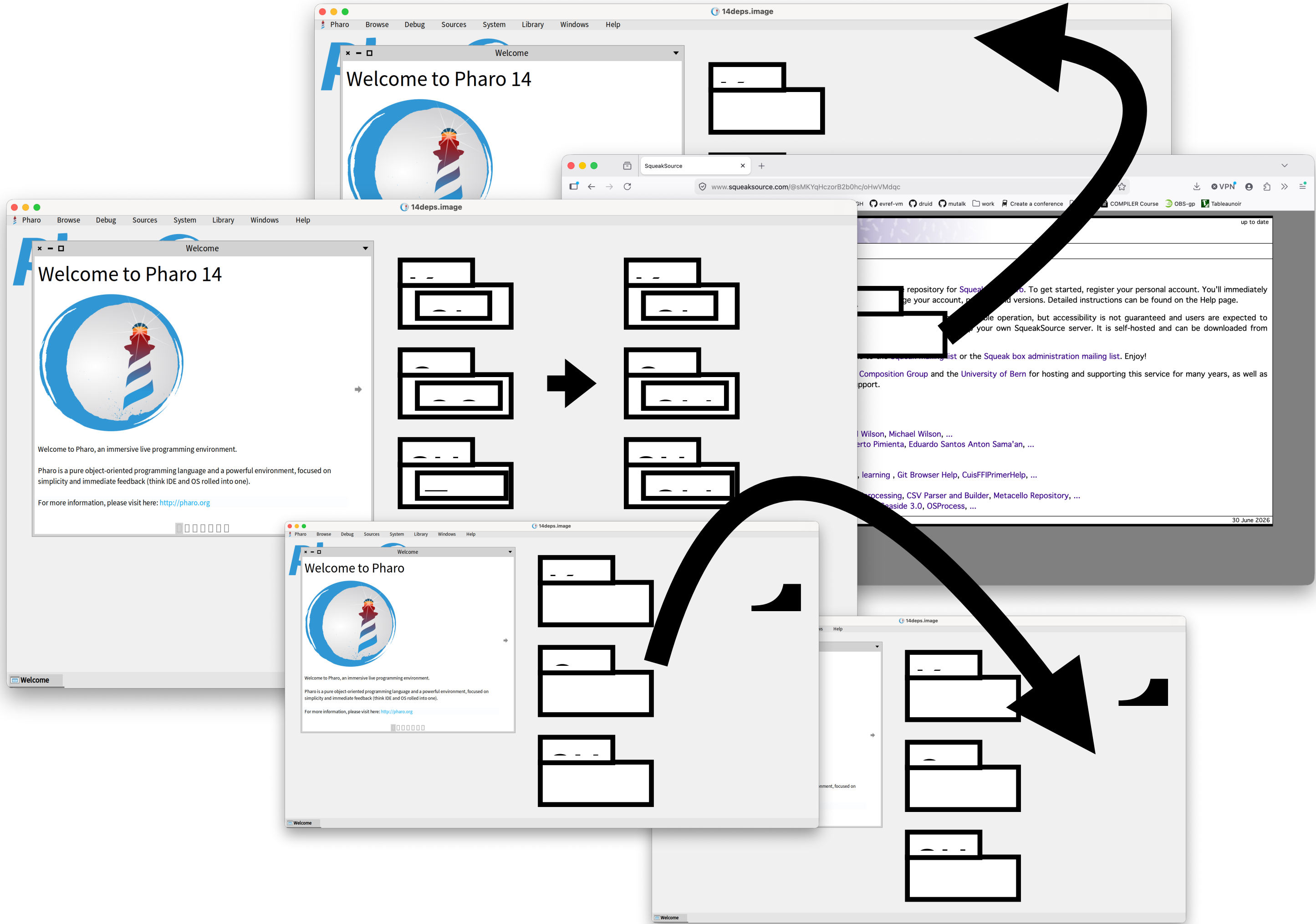


The Principle



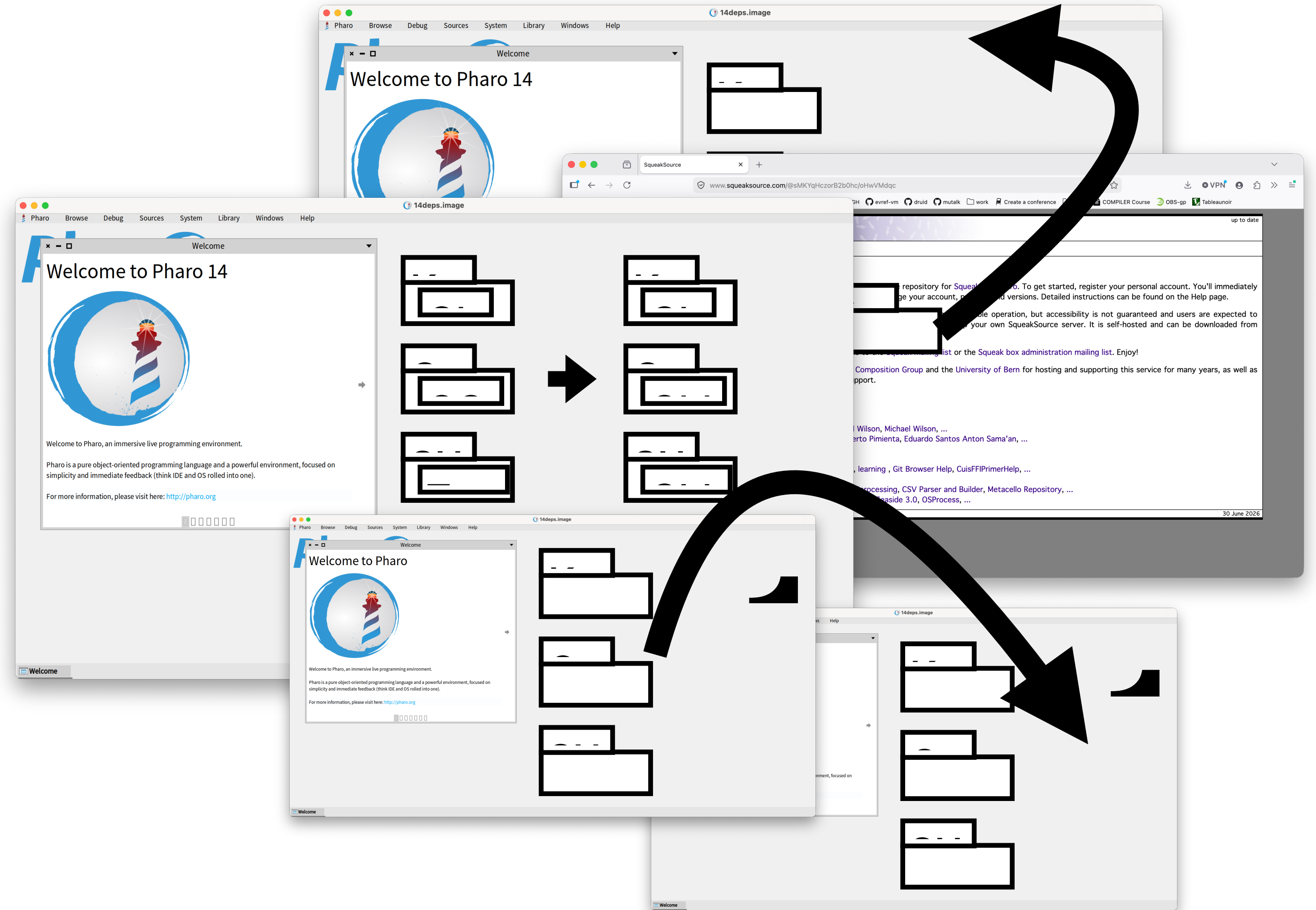
- Small namespaces
- Compiled separately
- Linked between them

Me, Writing *that* Module System



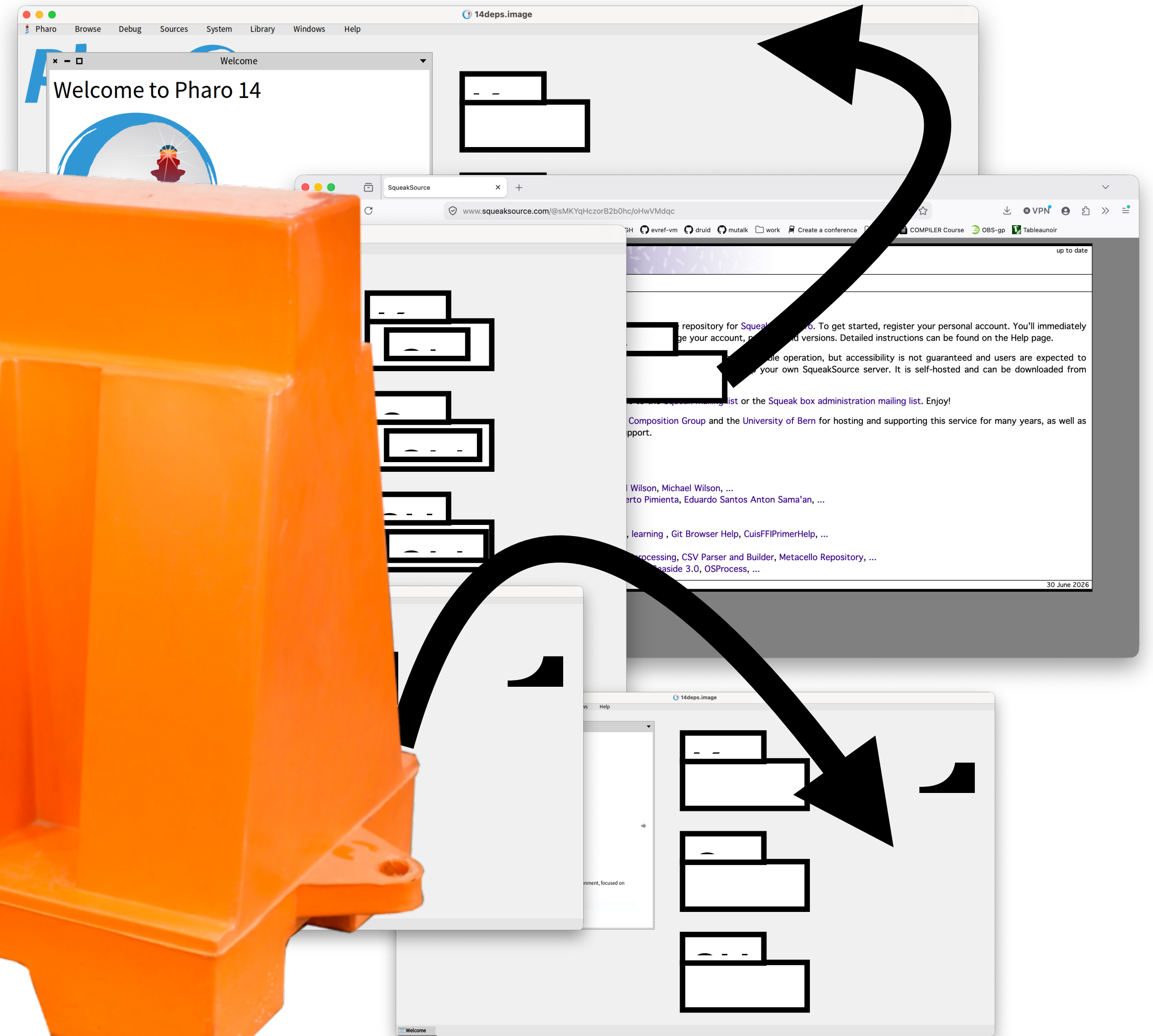
Me, Writing *that* Module System

- Import system
- Separate compilation
- Module registry



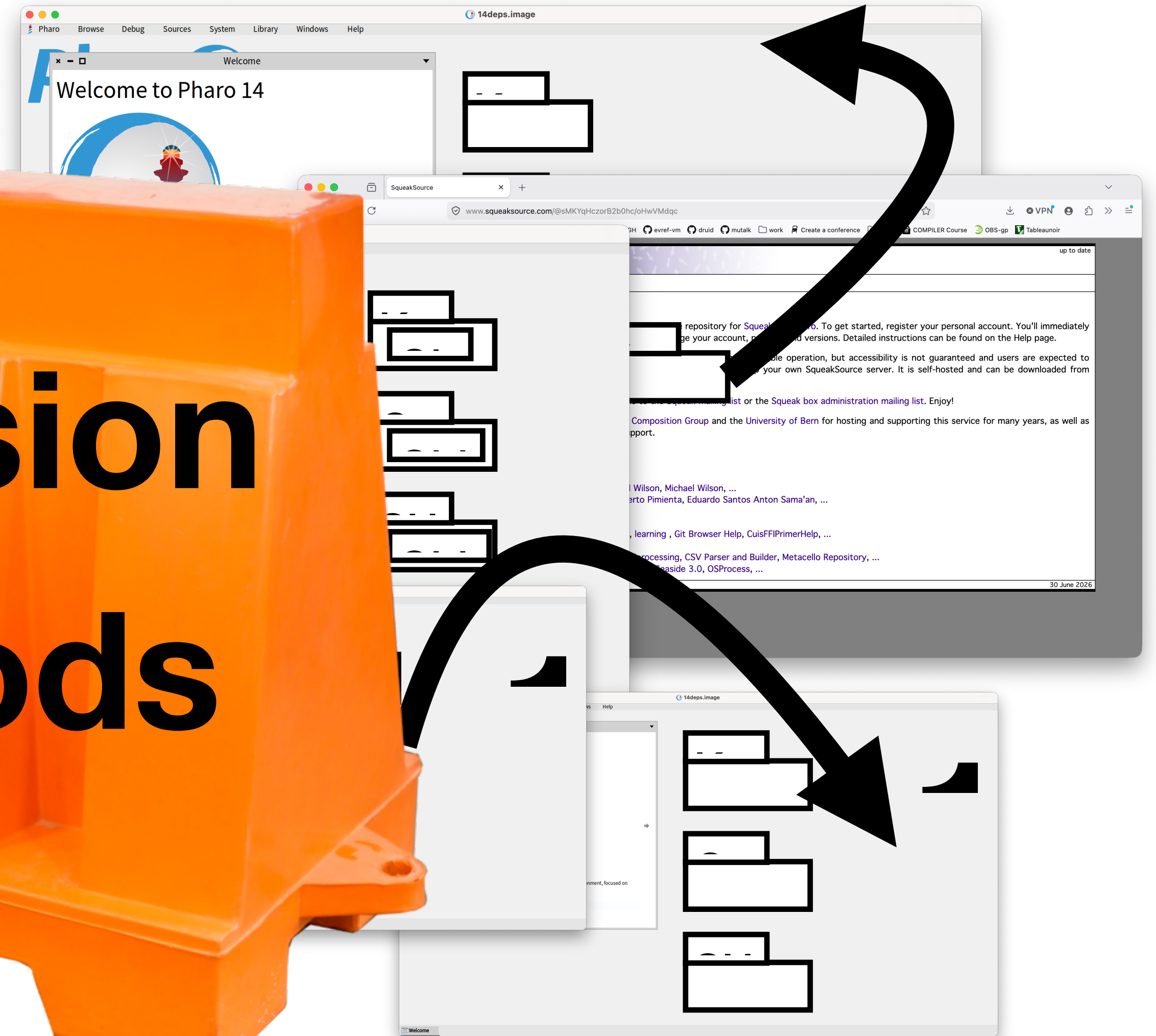
Me, Writing *that* Module System

- Import system
- Separate compilation
- Module registry



Me, Writing *that* Module System

- Import system
- Separate compilation
- Module registry



Extensions Break Modularity

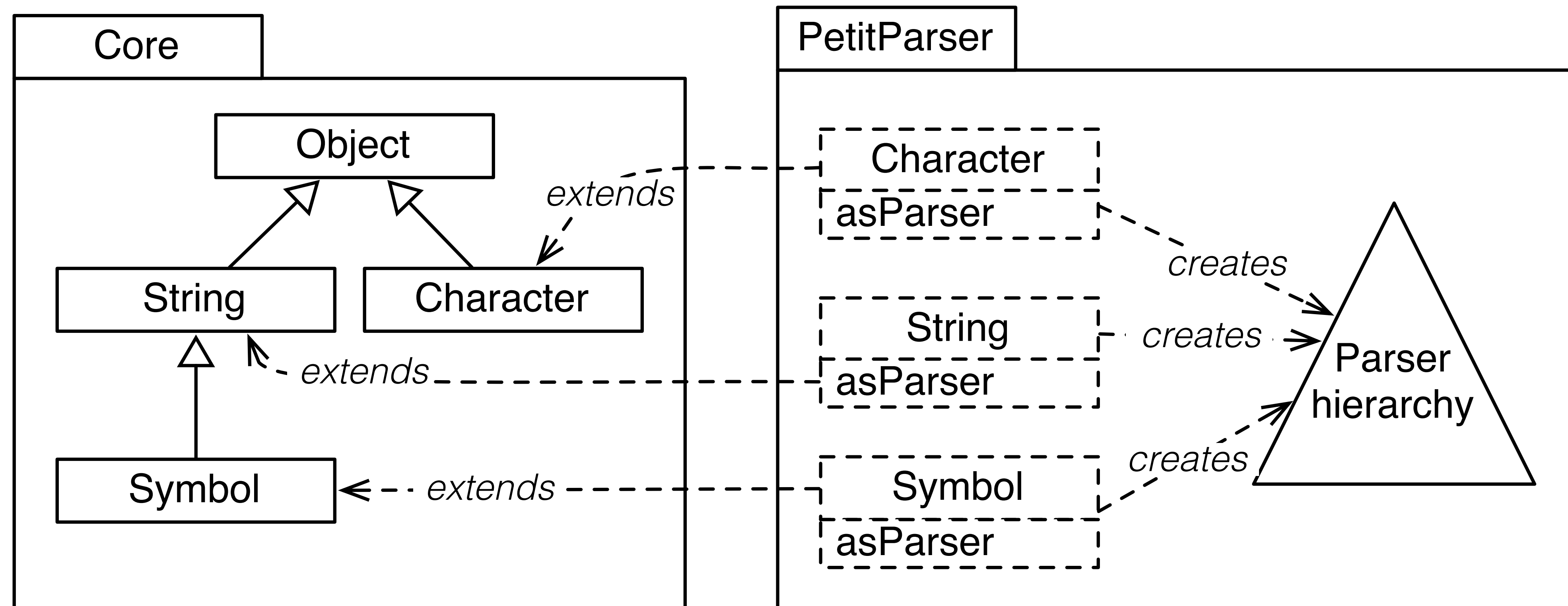
Extensions Break ~~Modularity~~

The Module System

Extension Methods



- Methods extending classes that belong to other packages



Extension Methods are useful



- Syntax sugar

```
PPStringParser on: aString => aString asParser
```

- Adapt interfaces

- and exploit polymorphism with classes we do not own

```
Parser >> asParser  
^ self
```

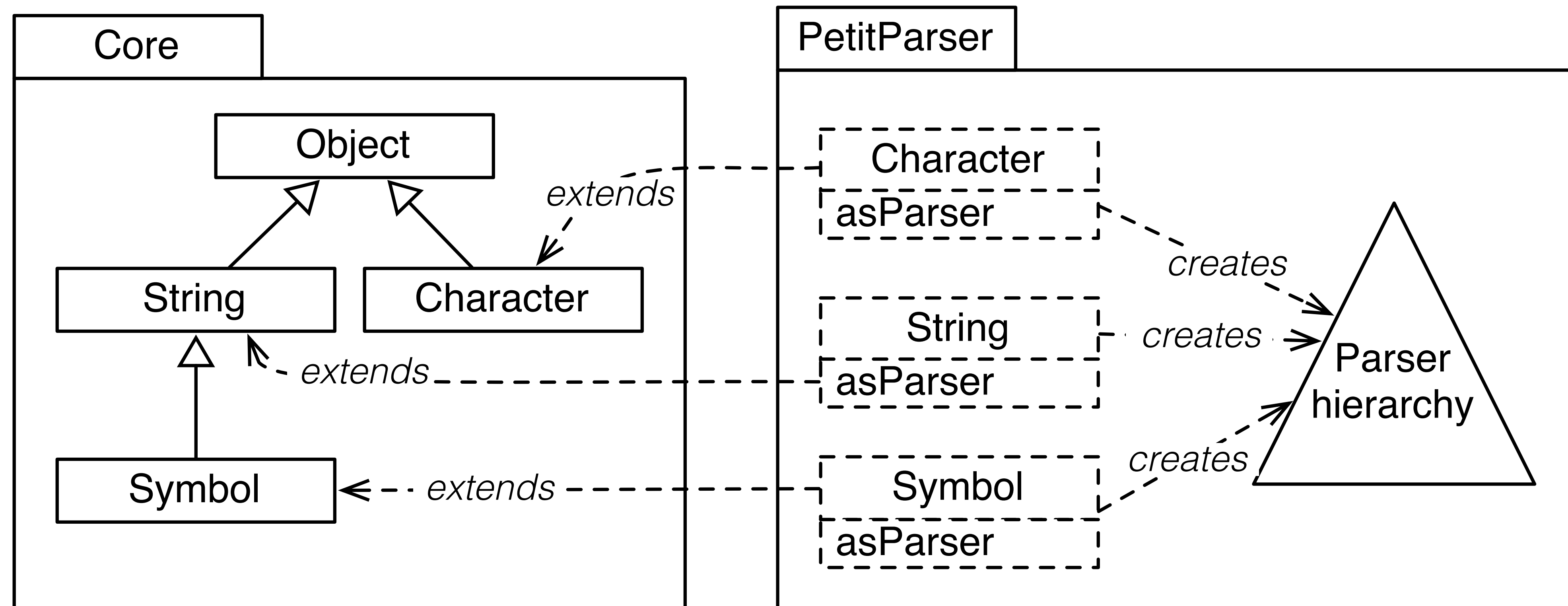
```
String >> asParser  
^ PPStringParser on: self
```

- Monkey patching

Extension Methods



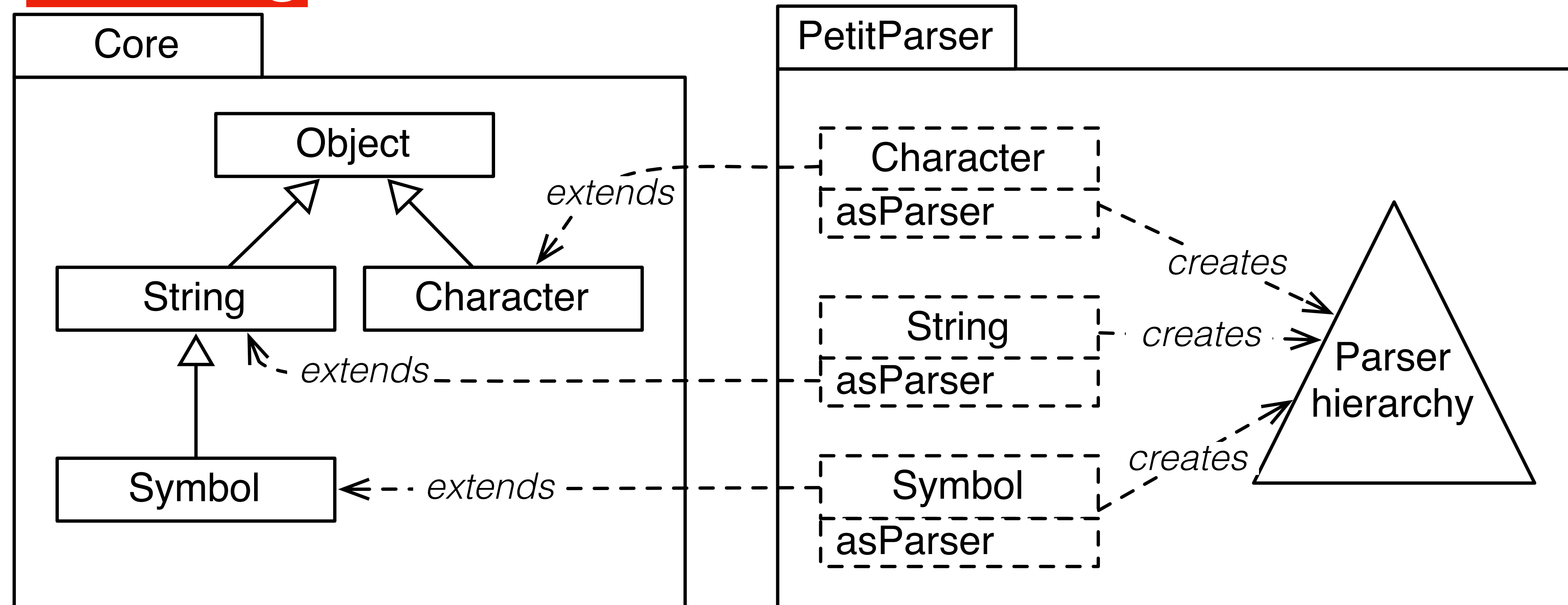
- Methods extending classes that **belong to other packages**



Why Extension Methods are EVIL



- Method **Maybe breaking** existing classes that **belong to other packages** **you don't own**



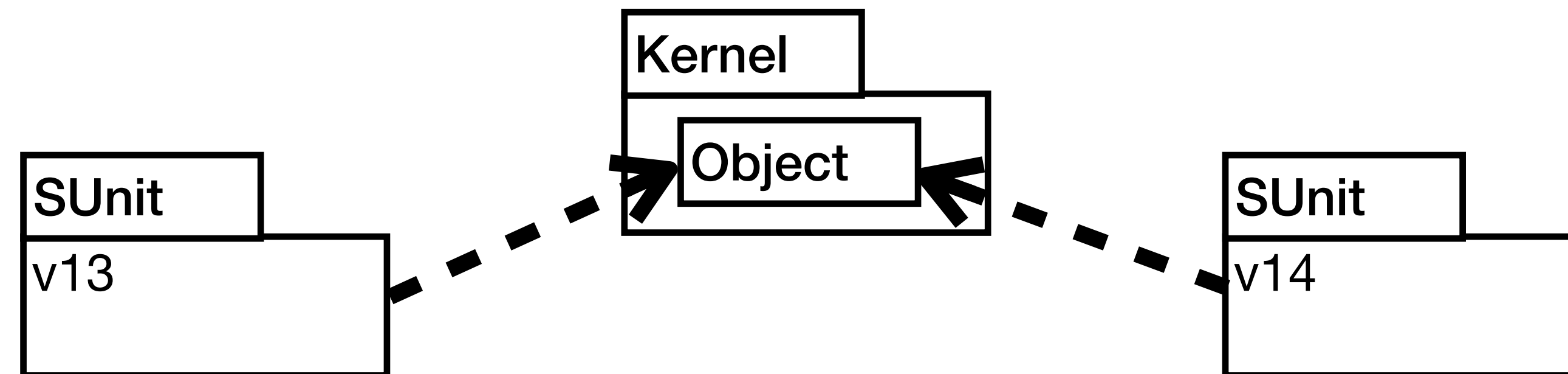
A Vision

Don't break
my toys,

I'll not break yours

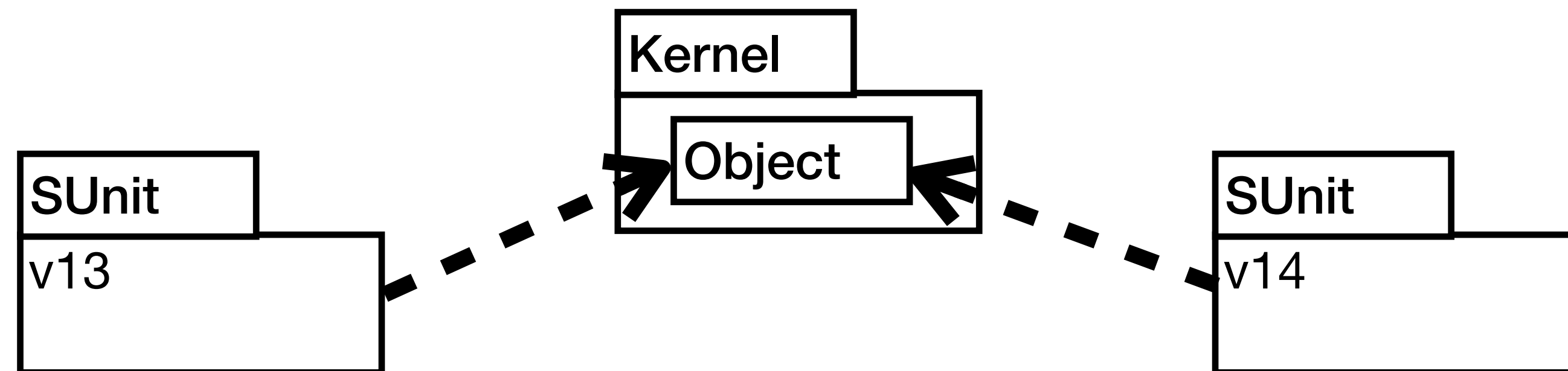
Global Extensions Prevent Multi-Versionning

- Two modules extend the same class with the same selector
- Conflict! Who wins?



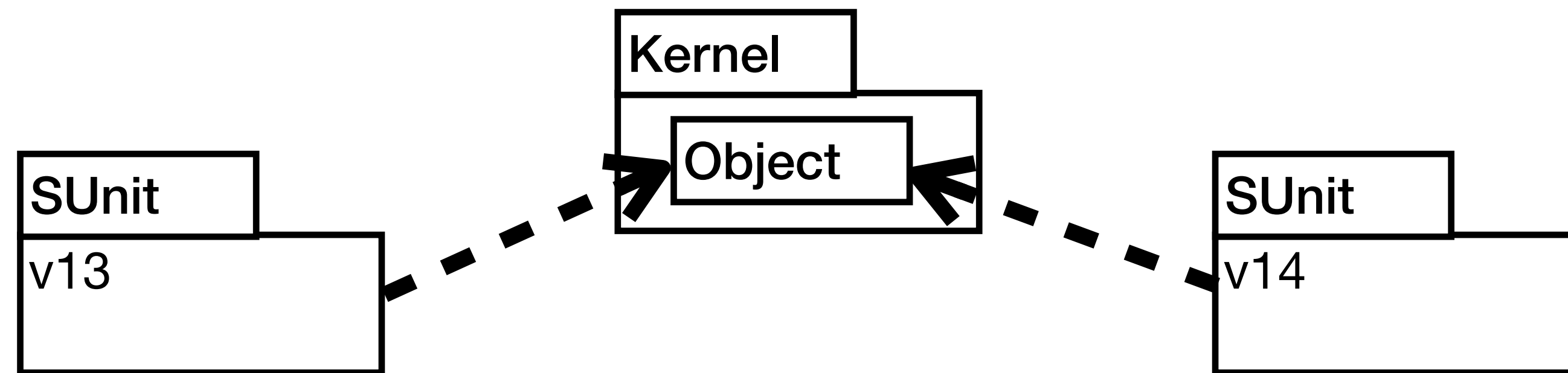
Global Extensions Prevent Multi-Versionning

- Two modules extend the same class with the same selector
- Conflict! Who wins?
- Nowadays, last one wins
- And **breaks** the first one



Global Extensions Prevent Multi-Versionning

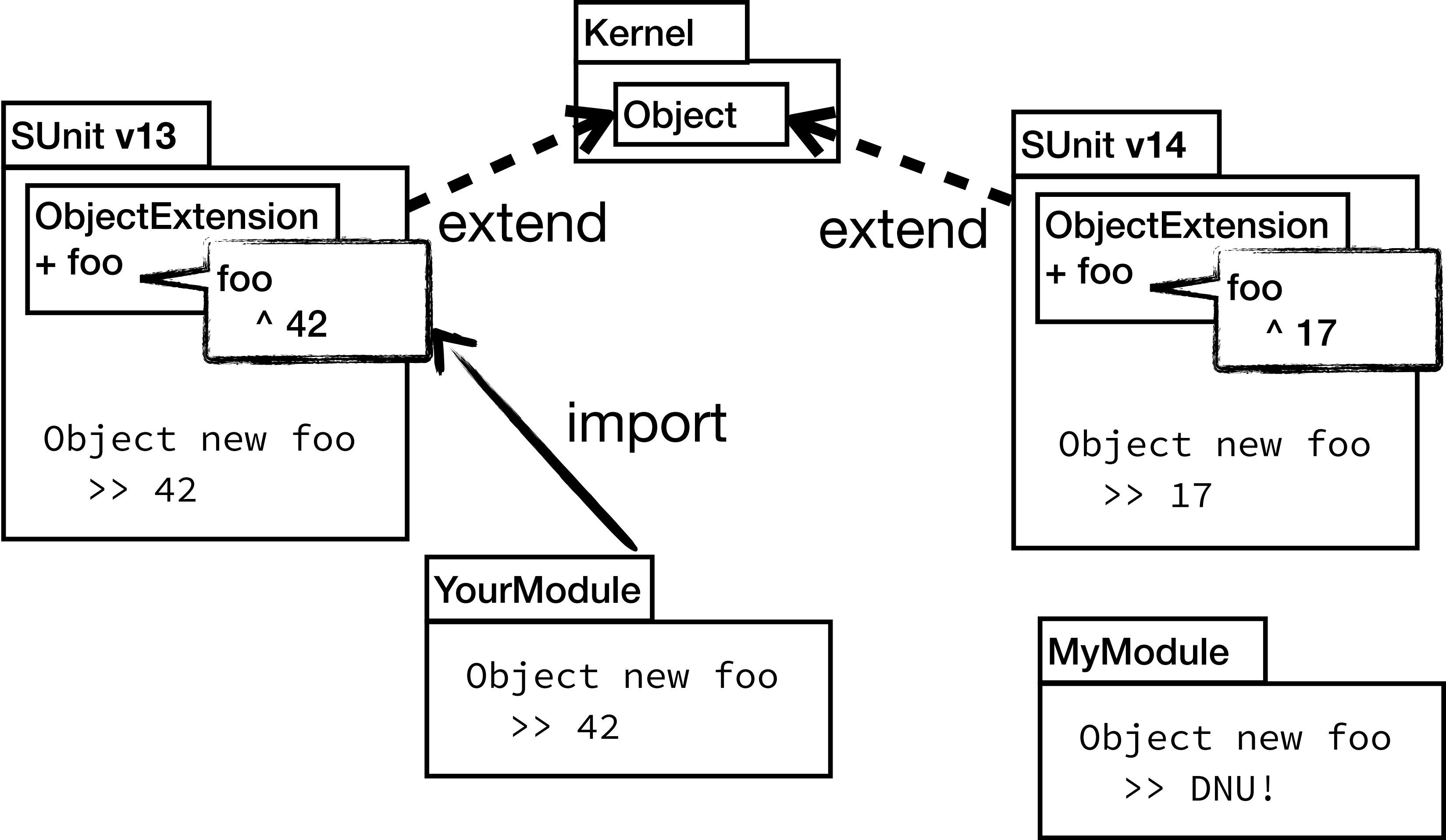
- Two modules extend the same class with the same selector
- Conflict! Who wins?
- Nowadays, last one wins
- And **breaks** the first one
- Better: **Scoping**
- Both win, in different contexts



Controlling Extension Methods

- New extension method model
- Key ideas:
 - You can extend, but it's not globally visible (think like protected/friends)
 - You will import extensions (like you import a class to subclass it)

Scoped Extension Methods

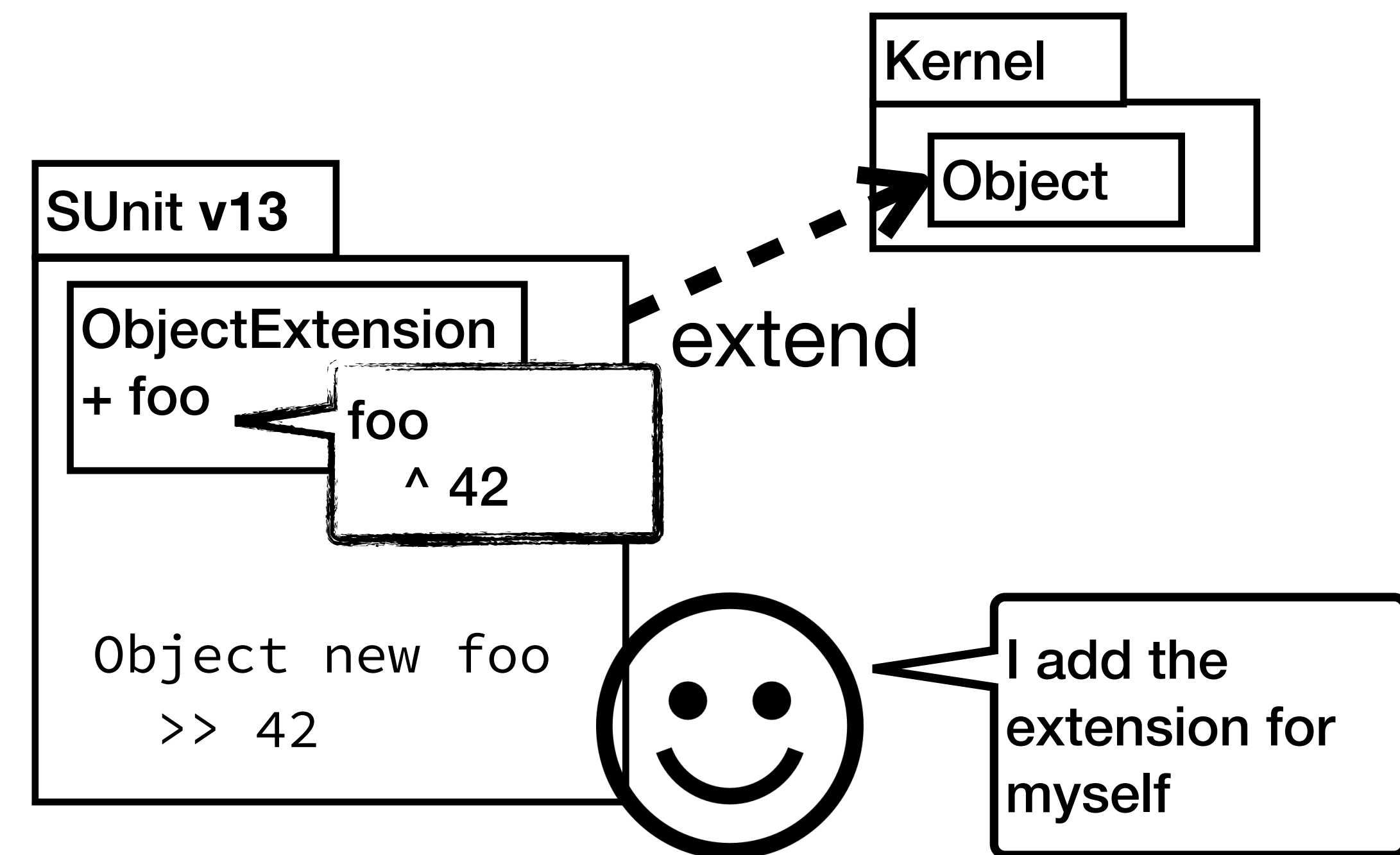


Sneak Peek at Extensions

- Pharo 14: 5055 extension methods, 3203 extension selectors
- ~1/2 methods are **easily scoped**: no redesign, minimal and automatable code rewrite

Lots of Extensions are *Local*

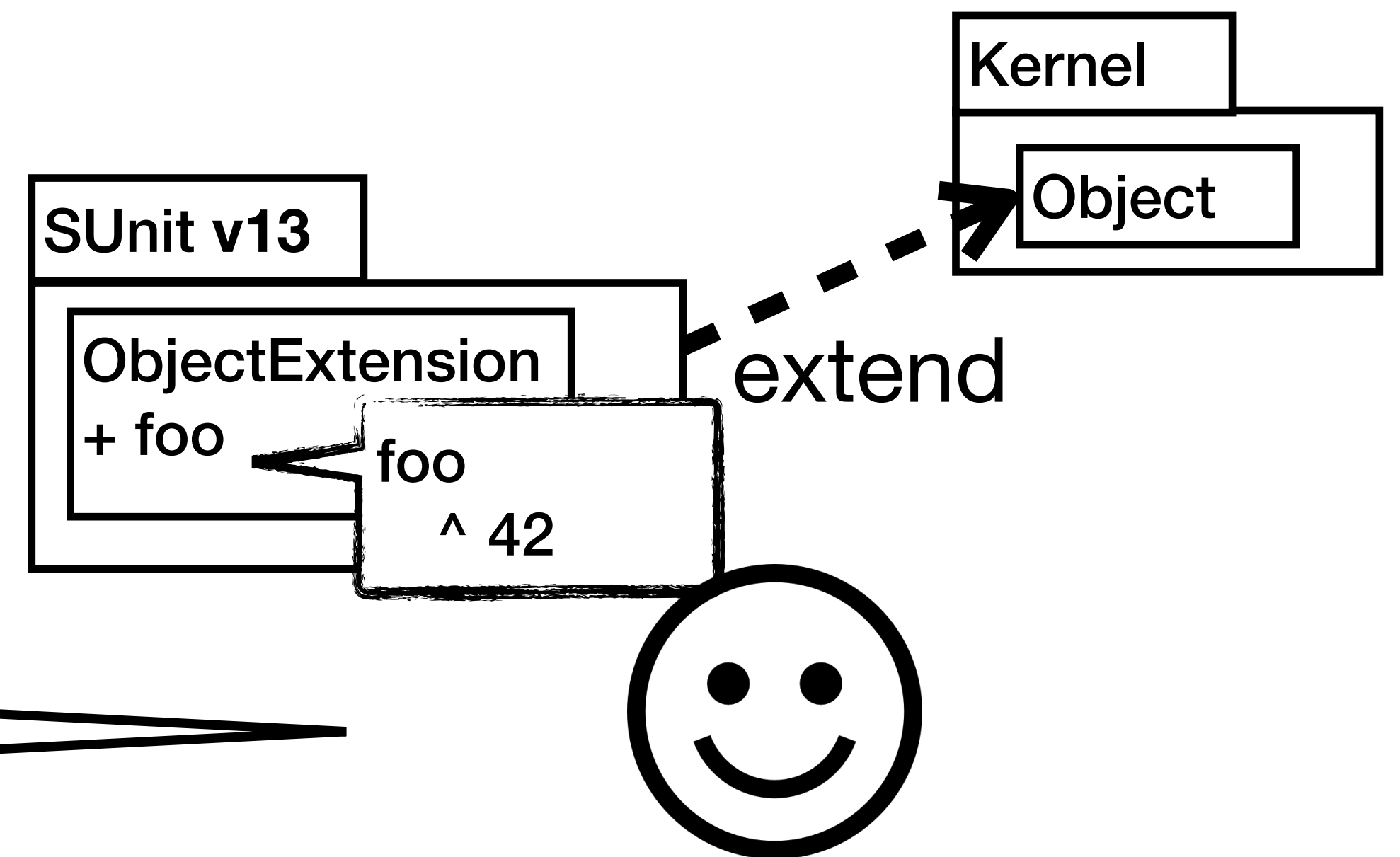
- 1680 (~1/3) are *local extensions*
- Defined by package A, used only by package A



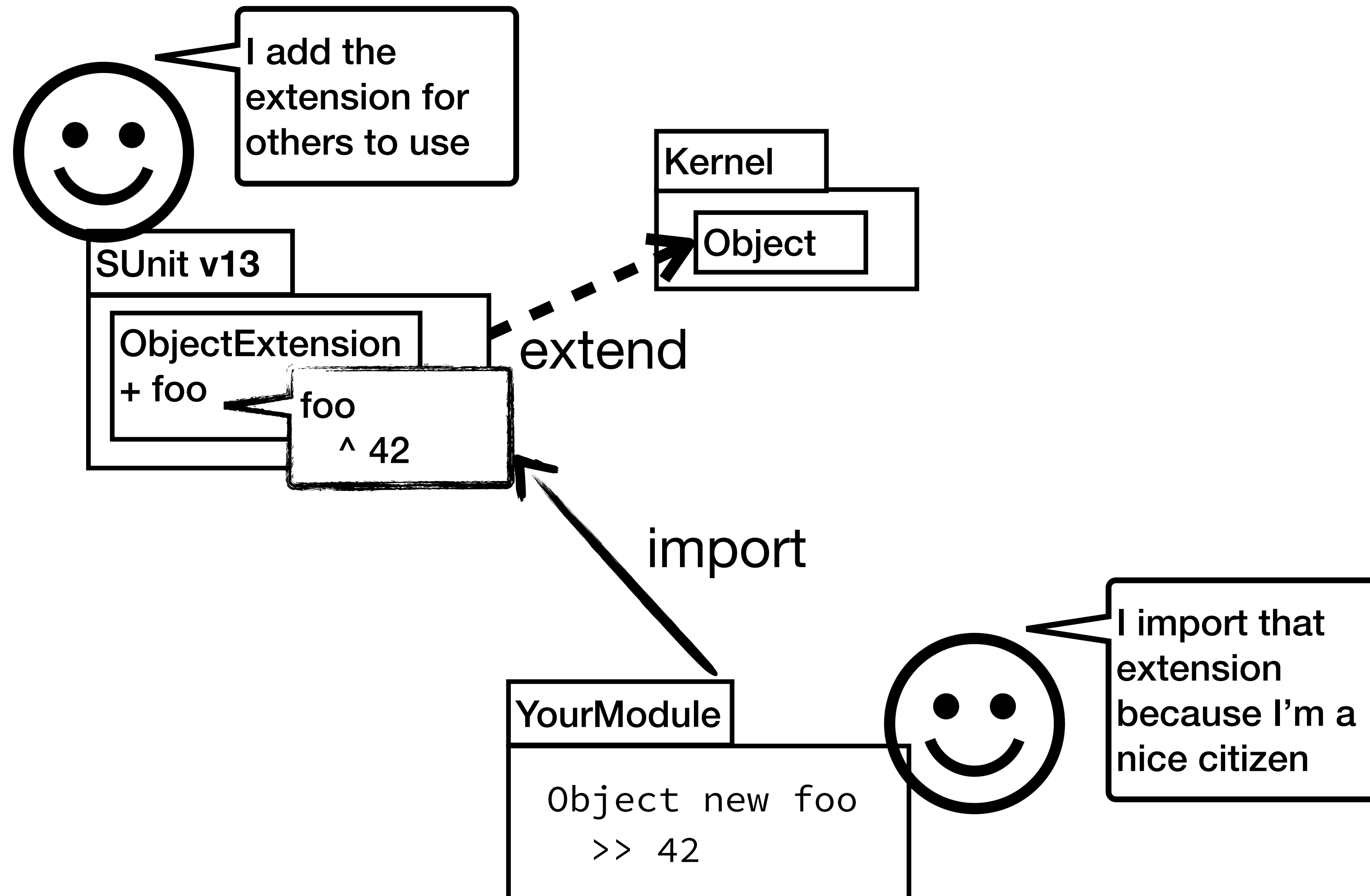
Lots of Extensions are *Monomorphic*

- 1645 (~1/3) are *static monomorphic extensions*
- Unique implementor, no polymorphism

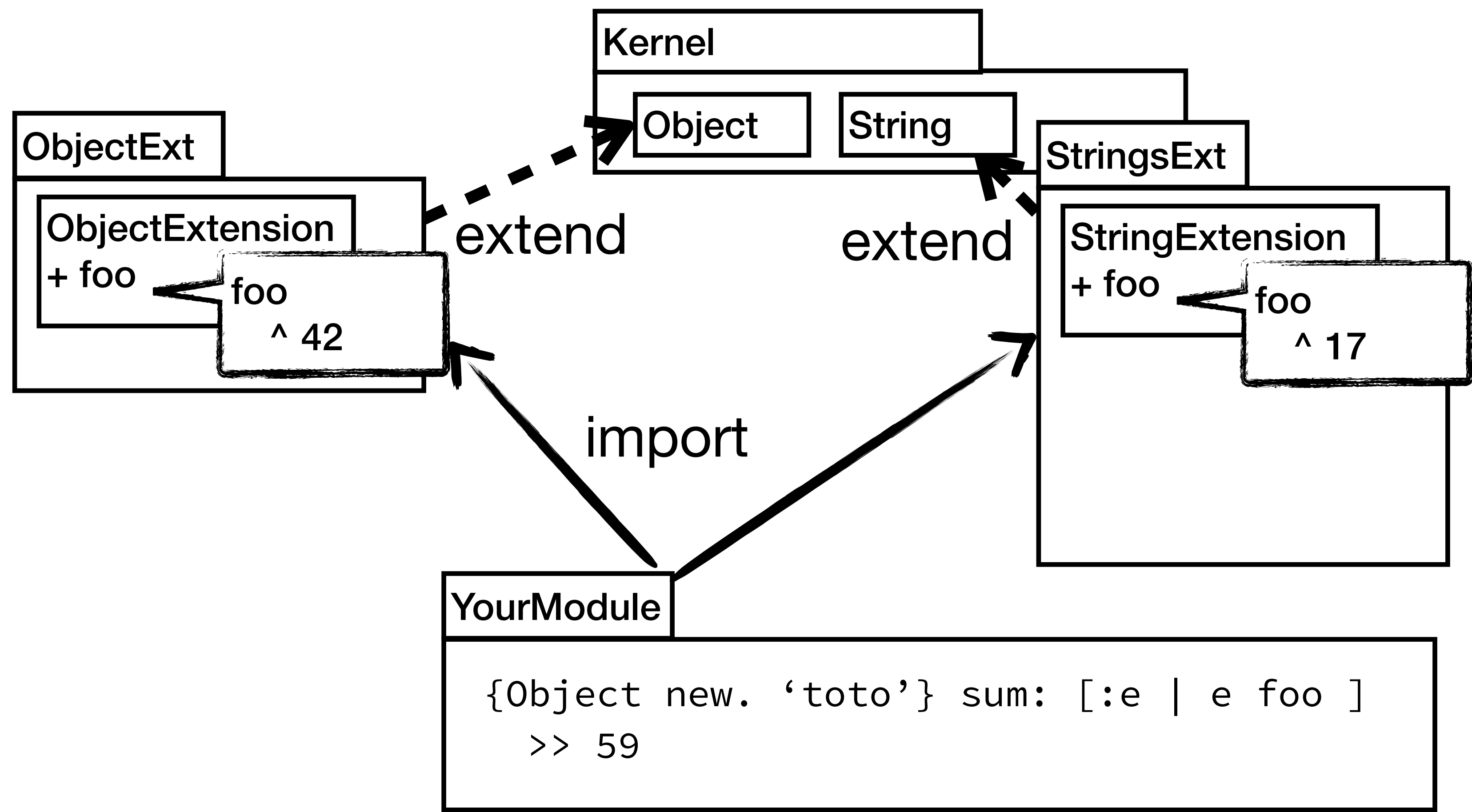
I'm the only implementor
of #foo.
If you want it, import it



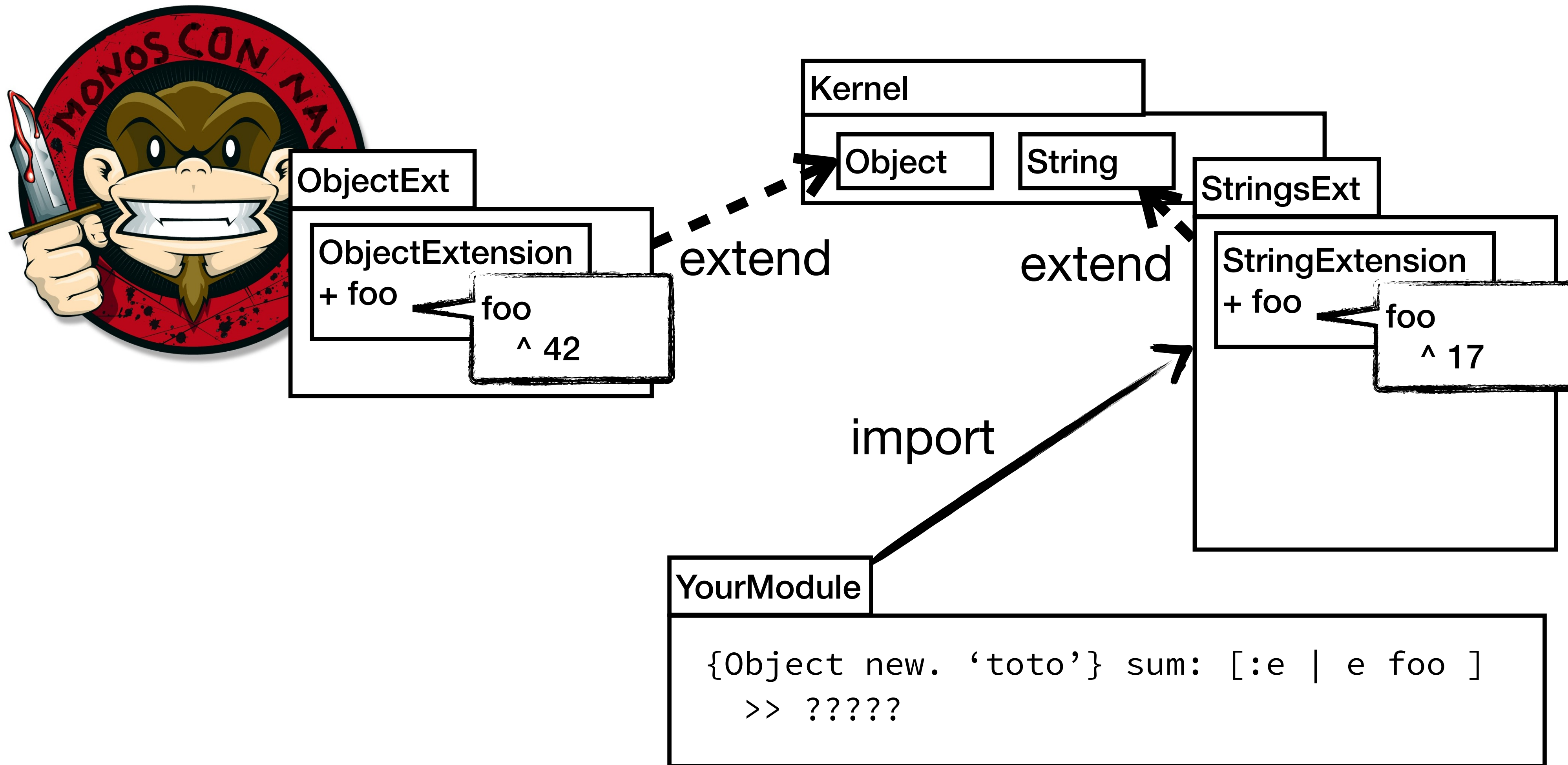
Lots of Other Good Extension Users



Polymorphism



The Bad Citizen: the Monkey Patcher



Identifying Border cases

- Key question: do you **own the *callsite***?
 - If you do, import the extension, and everything is **transparent**
 - If you don't, you're probably **patching code you don't own**
- **Border cases** will require slight rewriting/redesign

Identifying Border cases

Don't break
my toys,

I'll not break yours

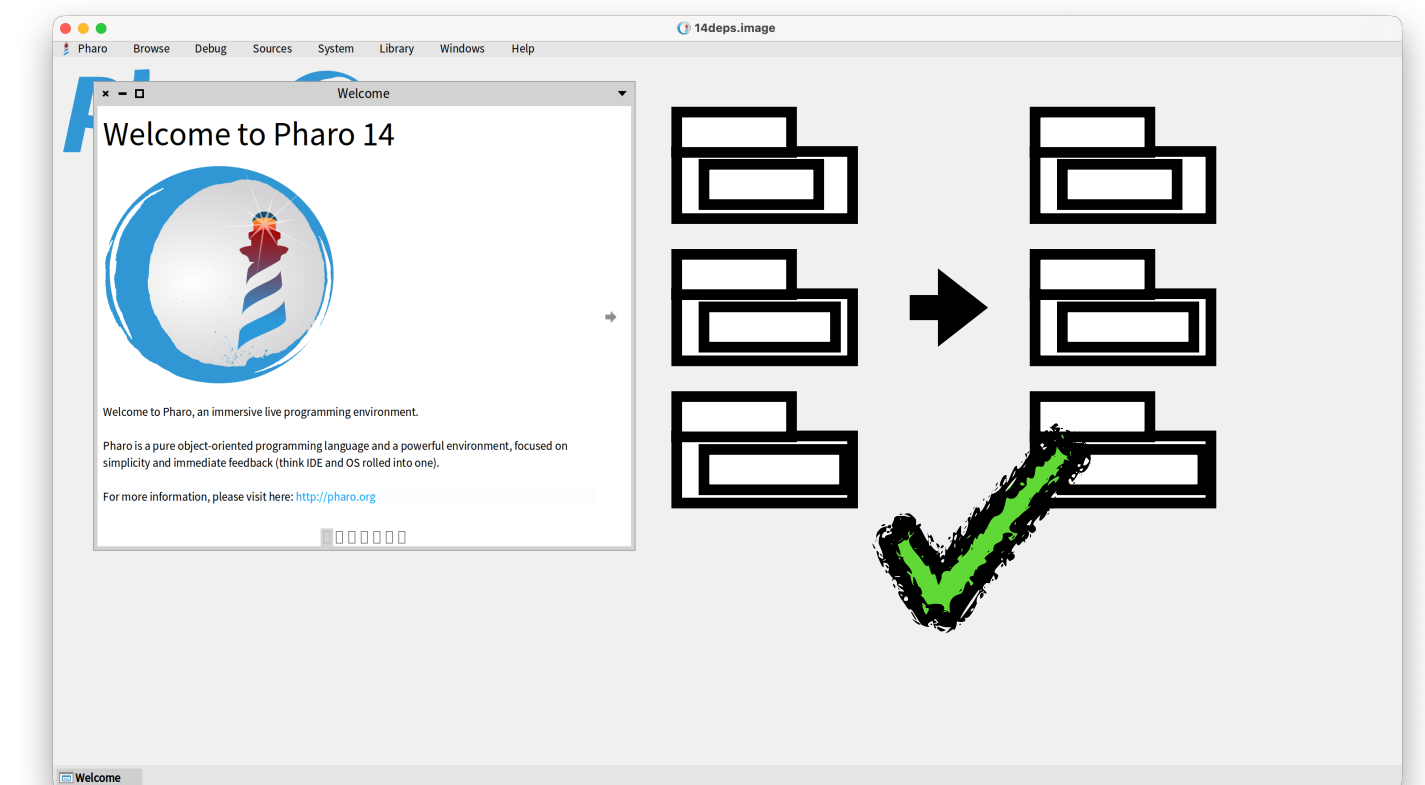
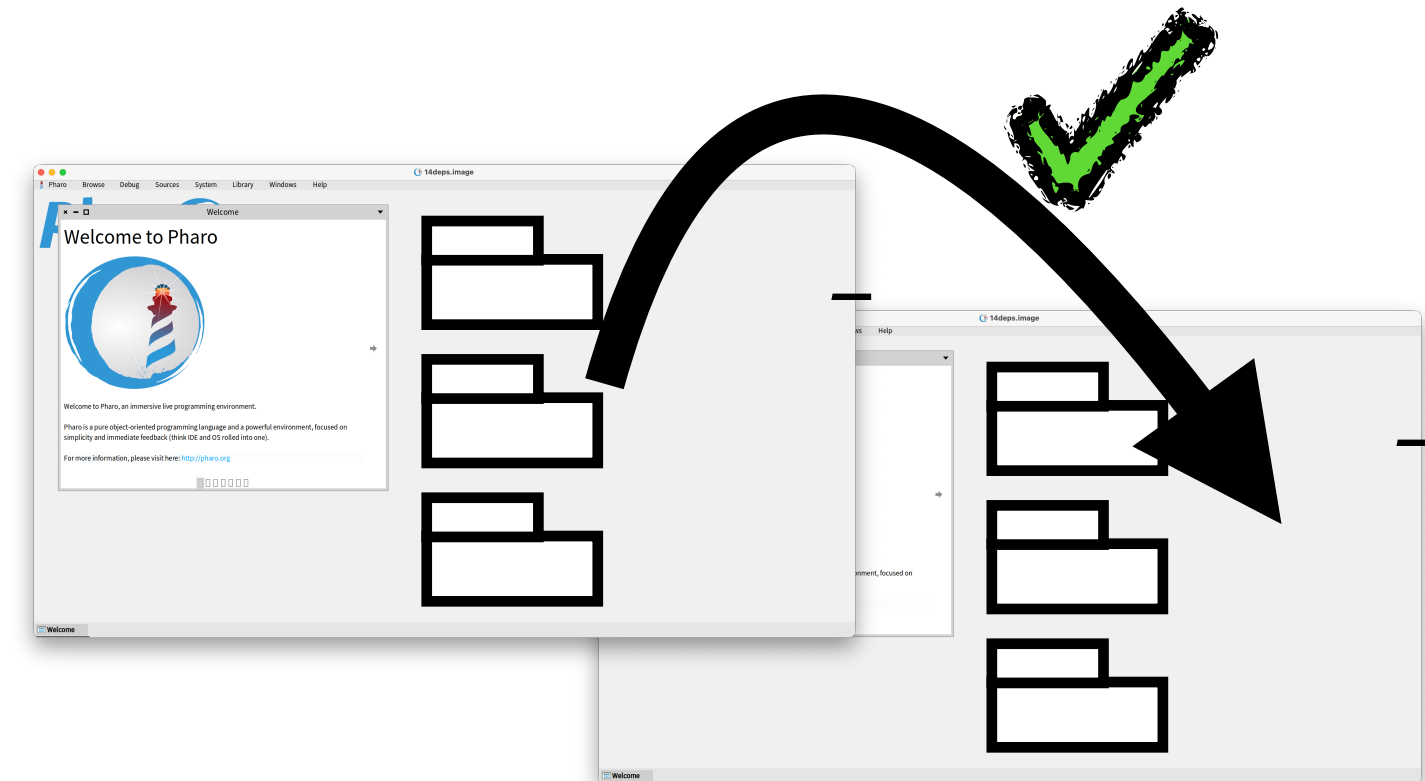
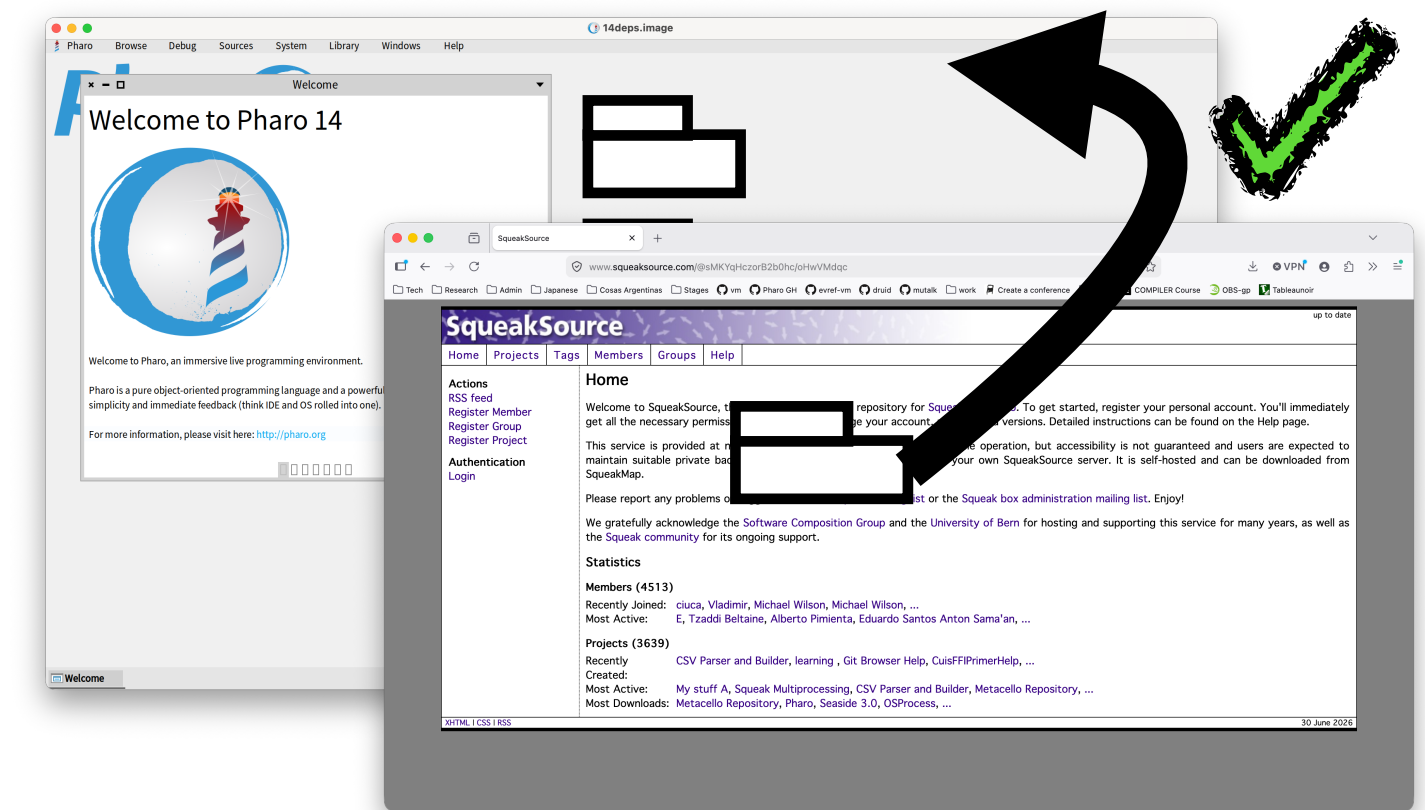
What's Done, What's Next?

- Module system + Separate compilation 
- PATH based import system 
- Scoped extensions with a coloring system 
- Module-aware Tools 
- Reflective access to classes! Lots of libraries assume single namespace :(
- Scoped extensions optimizations 

Modules = Namespaces ++

- = Name spaces + multi version + compatibility layers
- + scoped extensions
- As much **backward** compatible as possible
- Extra thanks to Luc, Stef, Nathan!

guillermo.polito@inria.fr
@guillep



Scoped Extension Methods

