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 *giʃe* (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

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

pharo-project / **pharo**

Code | Issues 1.4k | Pull requests 79 | Discussions | Act

Can we have namespaces? #1356

Closed

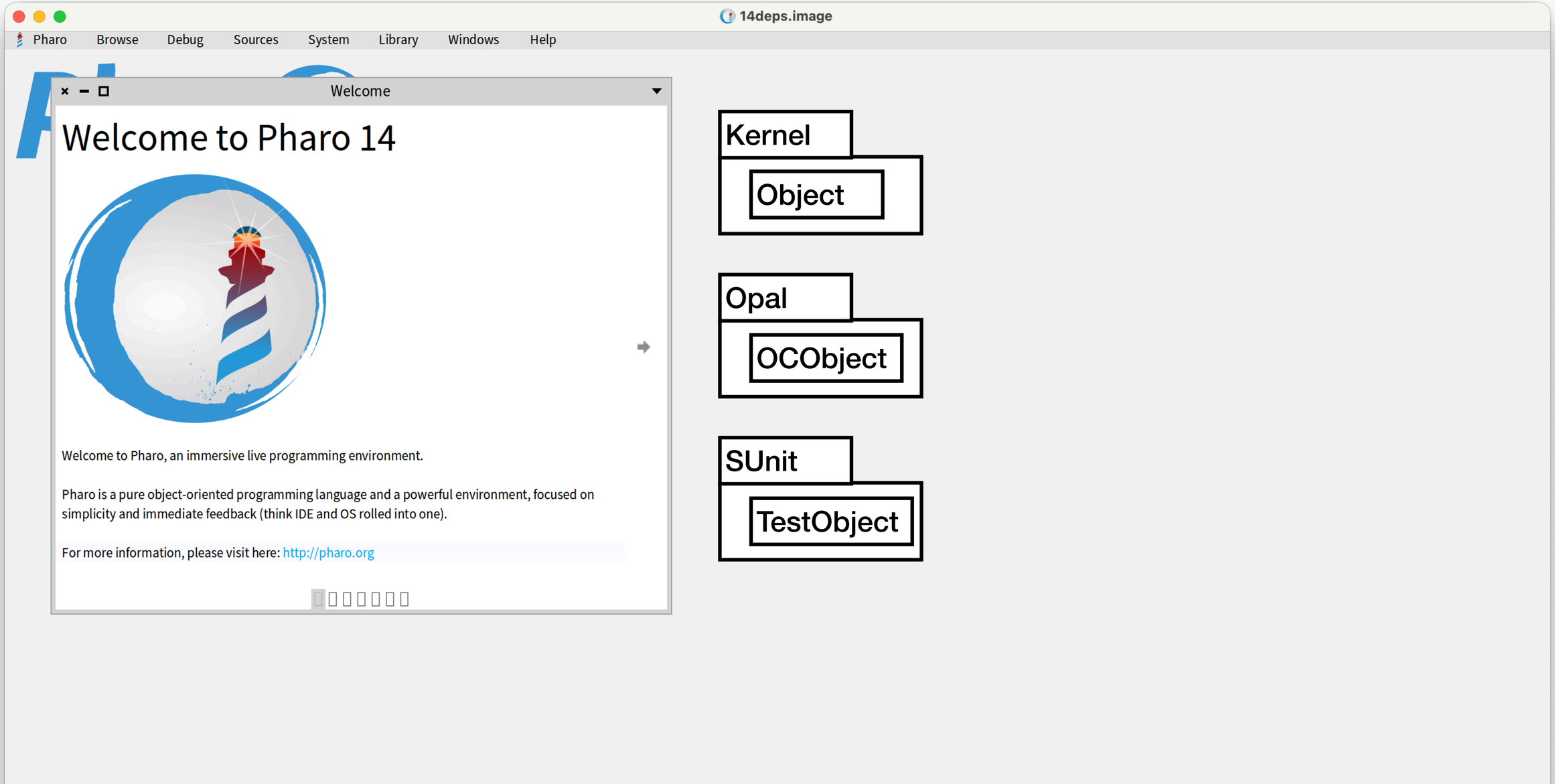
privat opened on Apr 26, 2023

Namespace are classical approach to deal with names conflicts caus
While the *two-uppercase-prefix* convention on classnames works loc

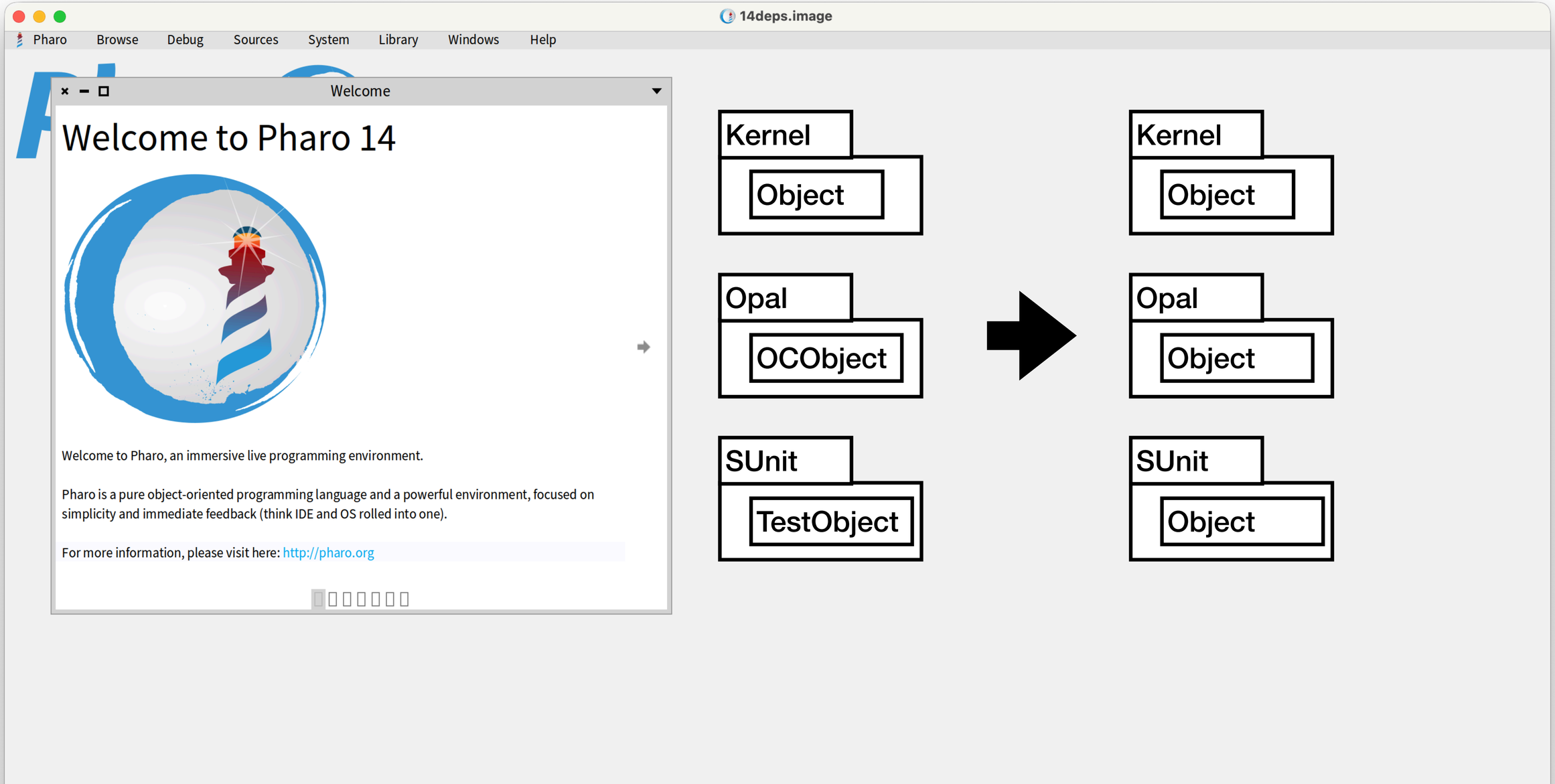
No

~~No~~
Namespaces are not enough

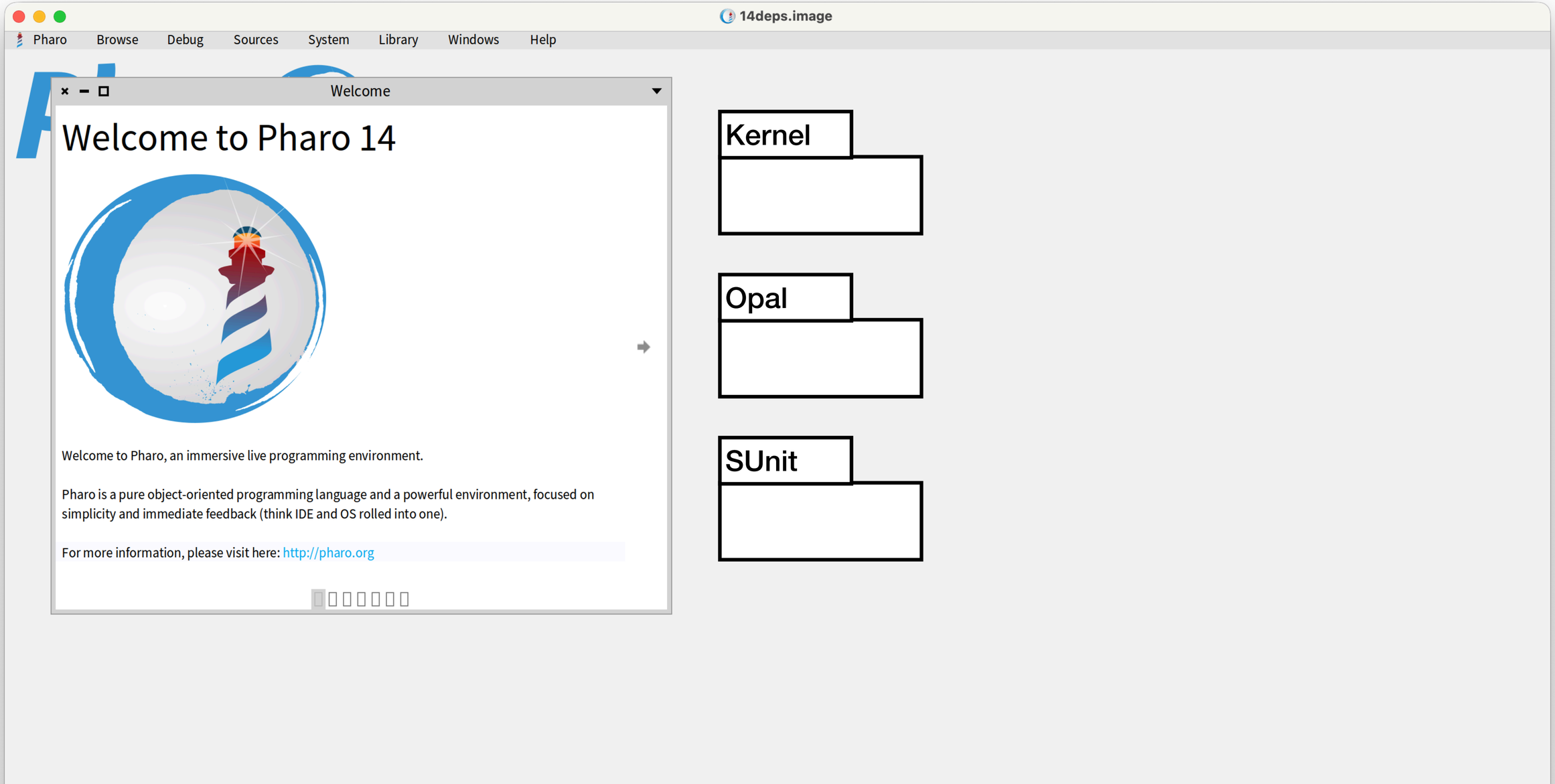
Namespaces = Name Spaces



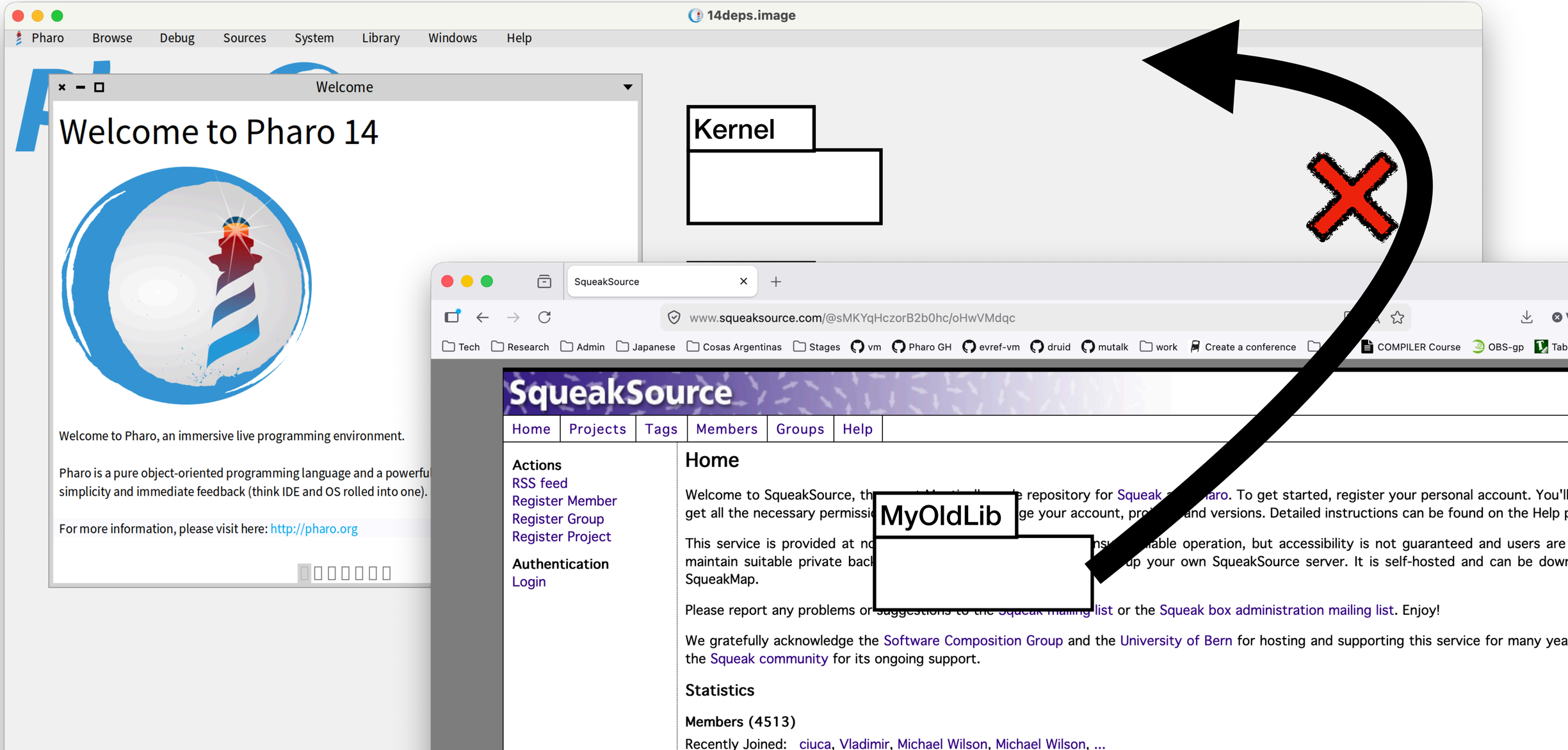
Namespaces closeTo: avoidPrefixes



Porting old code



Porting old code



Me @ the Future

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 Pharo Kernel (2016-17)
6 2016 vs 2026
7 Behind the Scenes
8 "Can we have namespaces?"
9 No
10 No
Namespaces are not enough
11 Pharo Single Namespace
12 Namespaces?

Modularity by Construction

Guille Polito

ESUG'26 - Plovdiv, Bulgaria

guillermo.polito@inria.fr
@guillep

Inria · Evref · cnrs · CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille · Université de Lille

1

Aspect
Titre
Corps
Numé
Arrière-pl
Star
Remplissa
Rempliss
Modifie

Me @ the Future Talking about Modularity

2026-ESUG-Modularity

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

Bootstrap

Refactoring

Code Migration

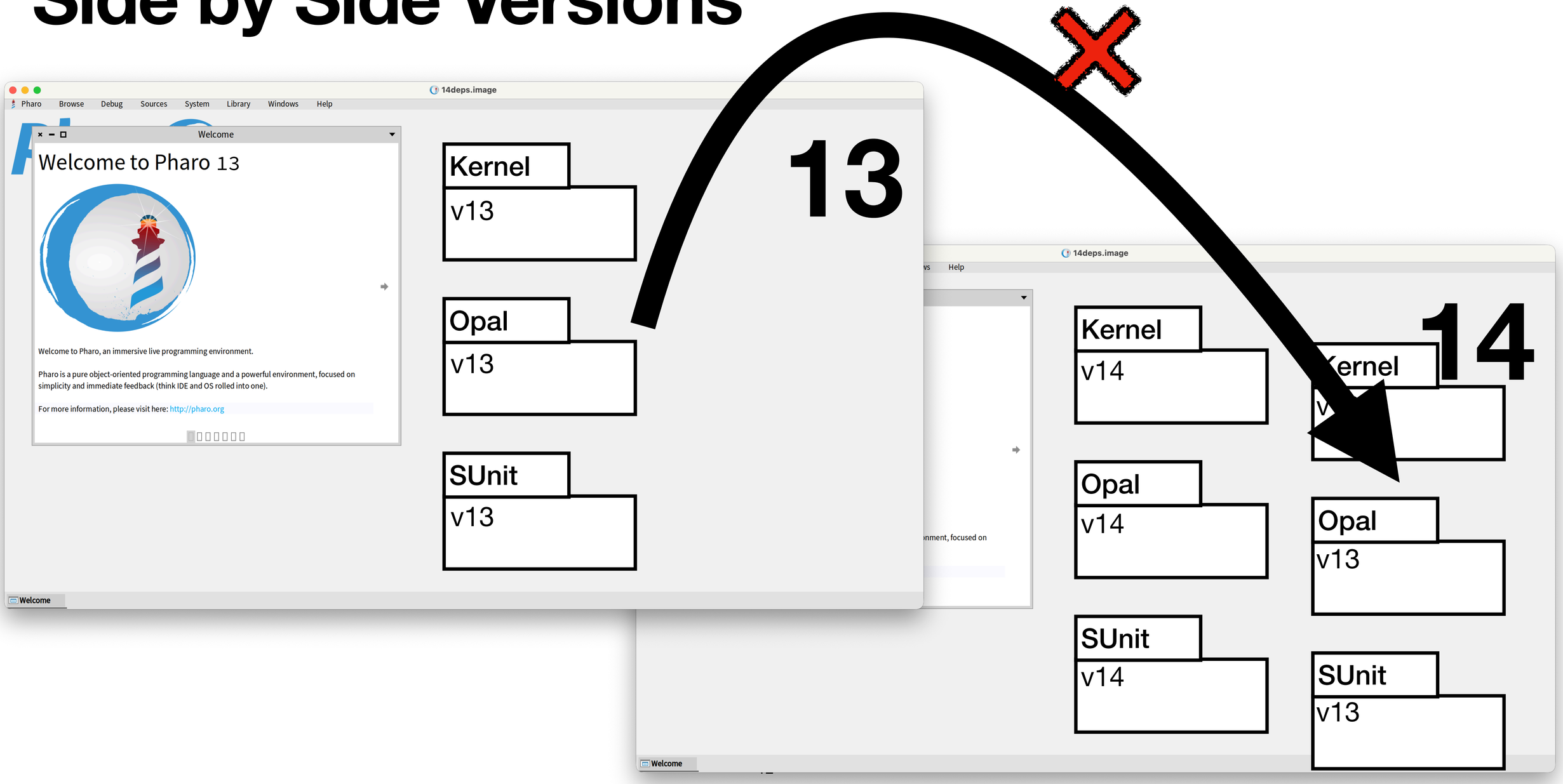
Compatibility

Modularity by Construct

ESUG'26 - Plovdiv, Bulgaria

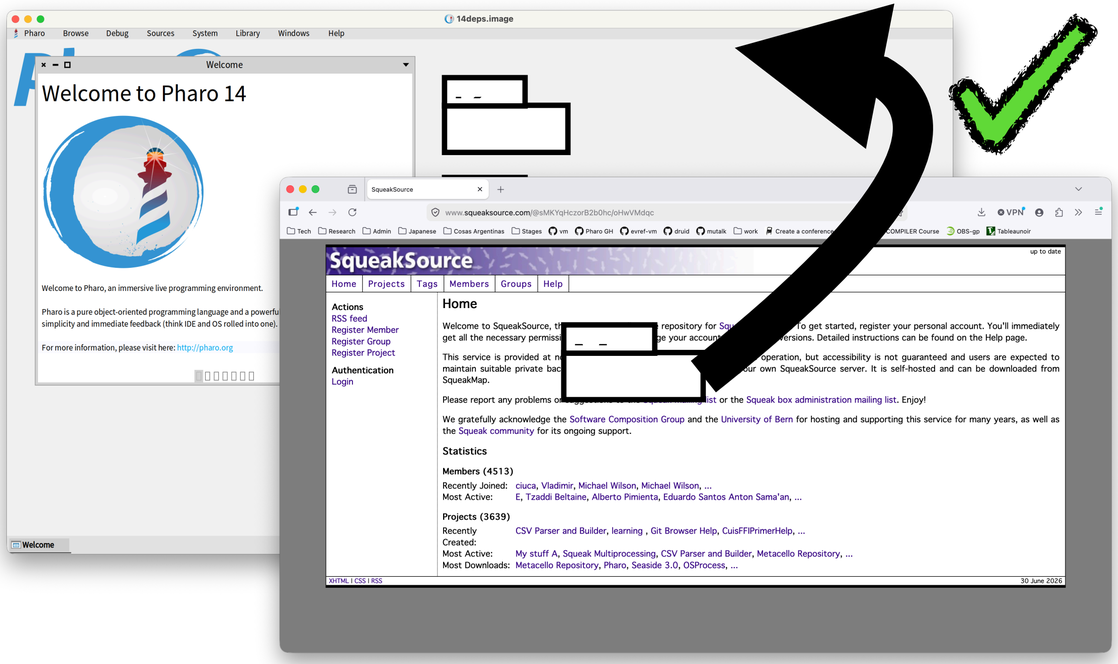
Inria · Evref · cnrs · CRIStAL · Université de Lille

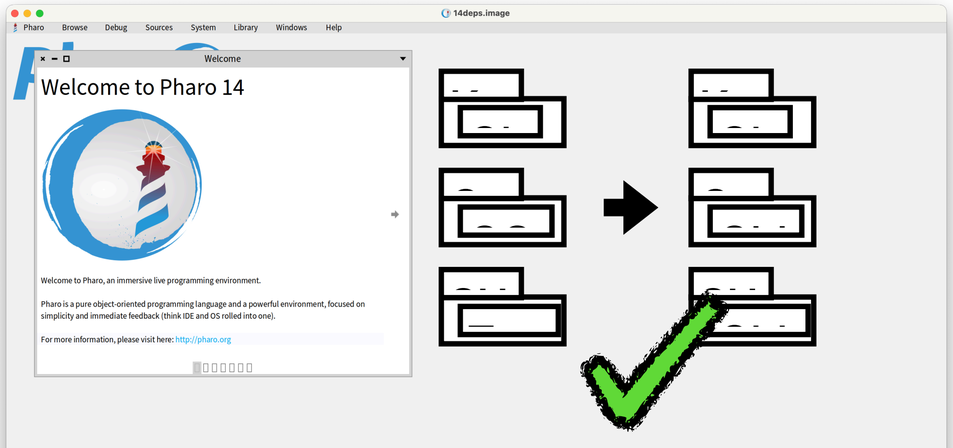
Side by Side Versions

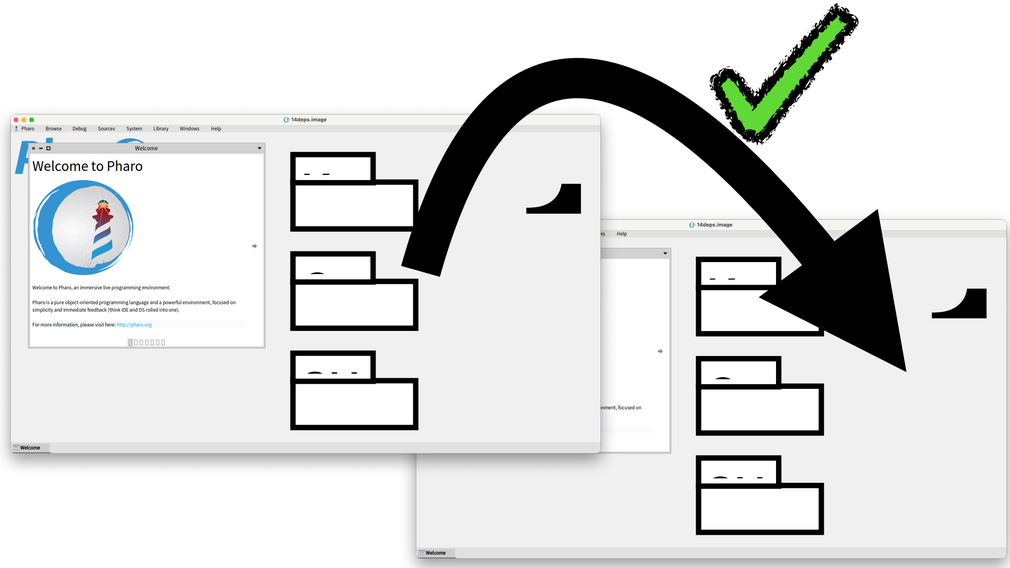


Goal

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



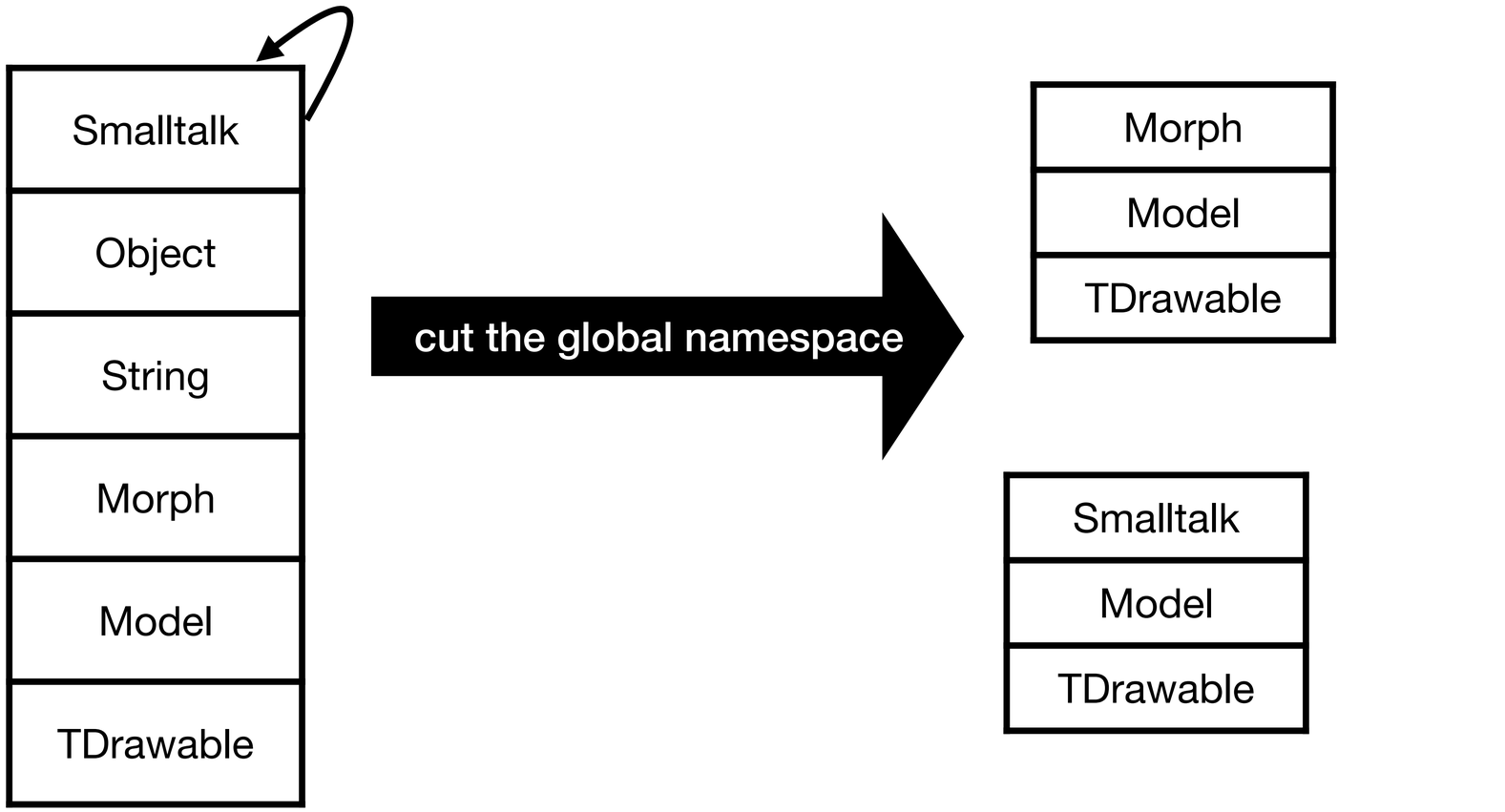




Goal

- Name s
- Ease of
- Simplify
- As muc

The Principle

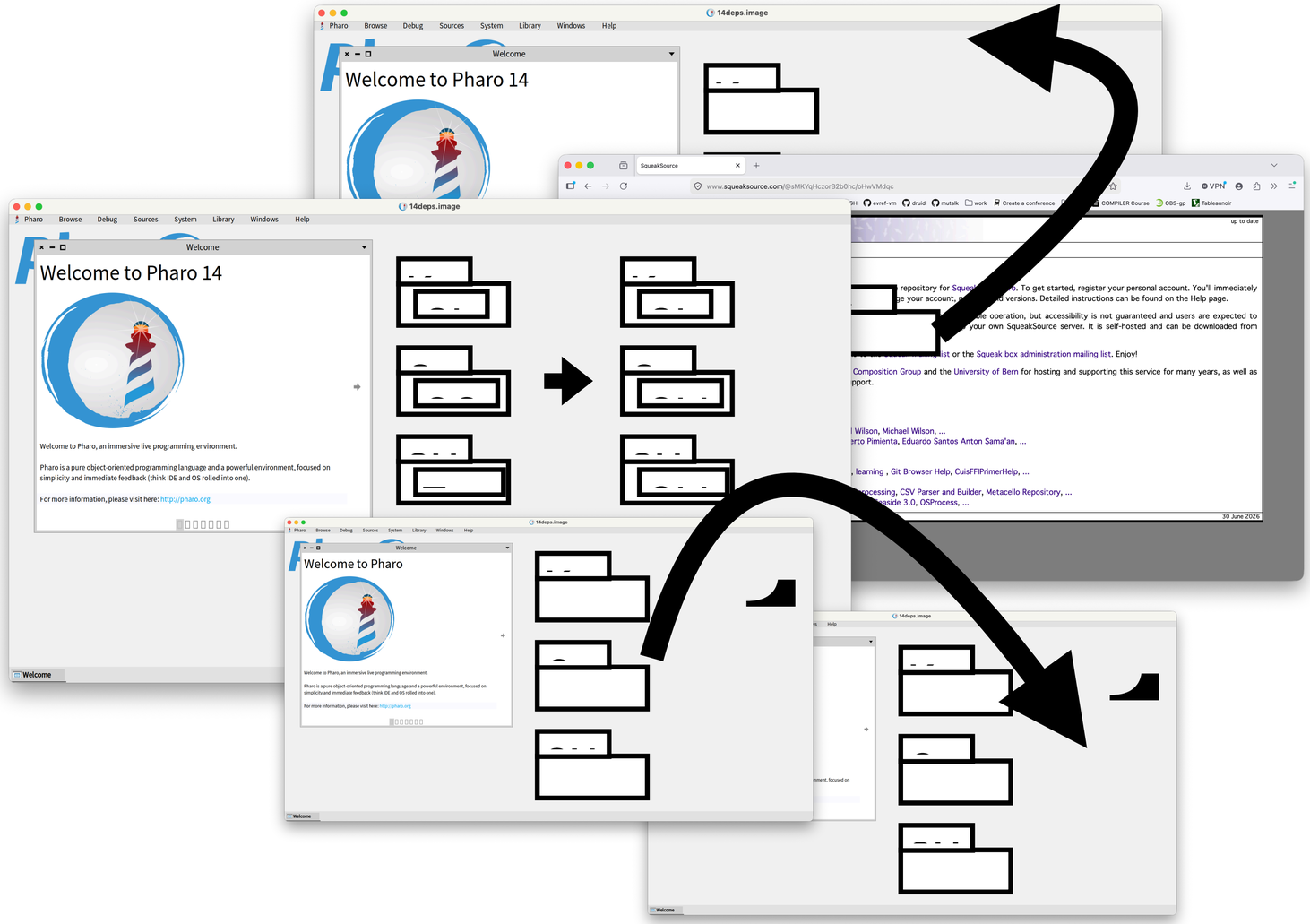


`Smalltalk globals`

~~`Smalltalk globals`~~

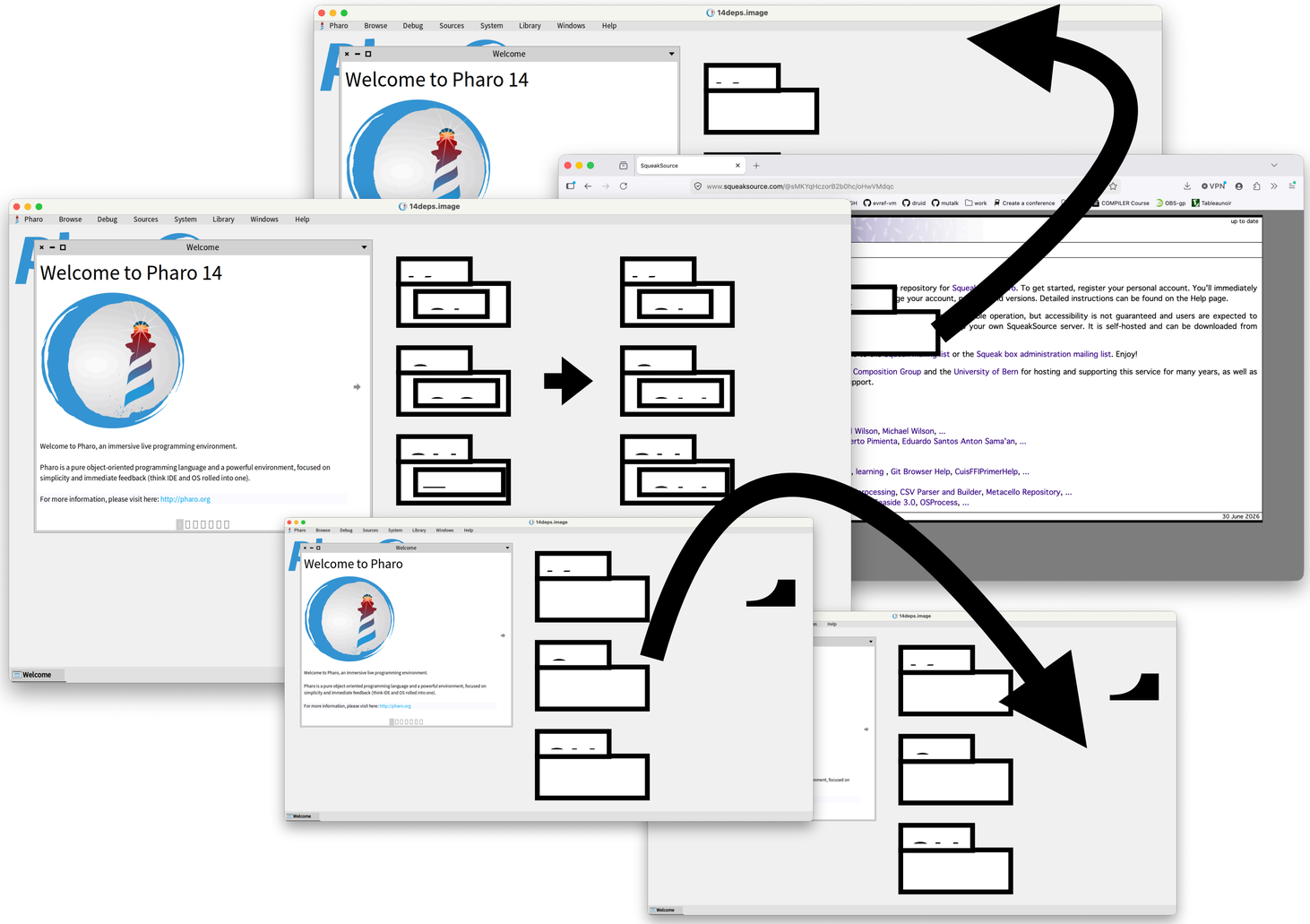
- Small namespaces
- Compiled separately
- Linked between them

Me, Writing *that* Module System



Me, Writing *that* Module System

- Import system
- Separate compilation
- Module registry



Me, Writing *that* Module System

- Import system
- Separate compilatio
- Module registry

Me, Writing *that* Module System

- Import system
- Separate compilatio
- Module registry

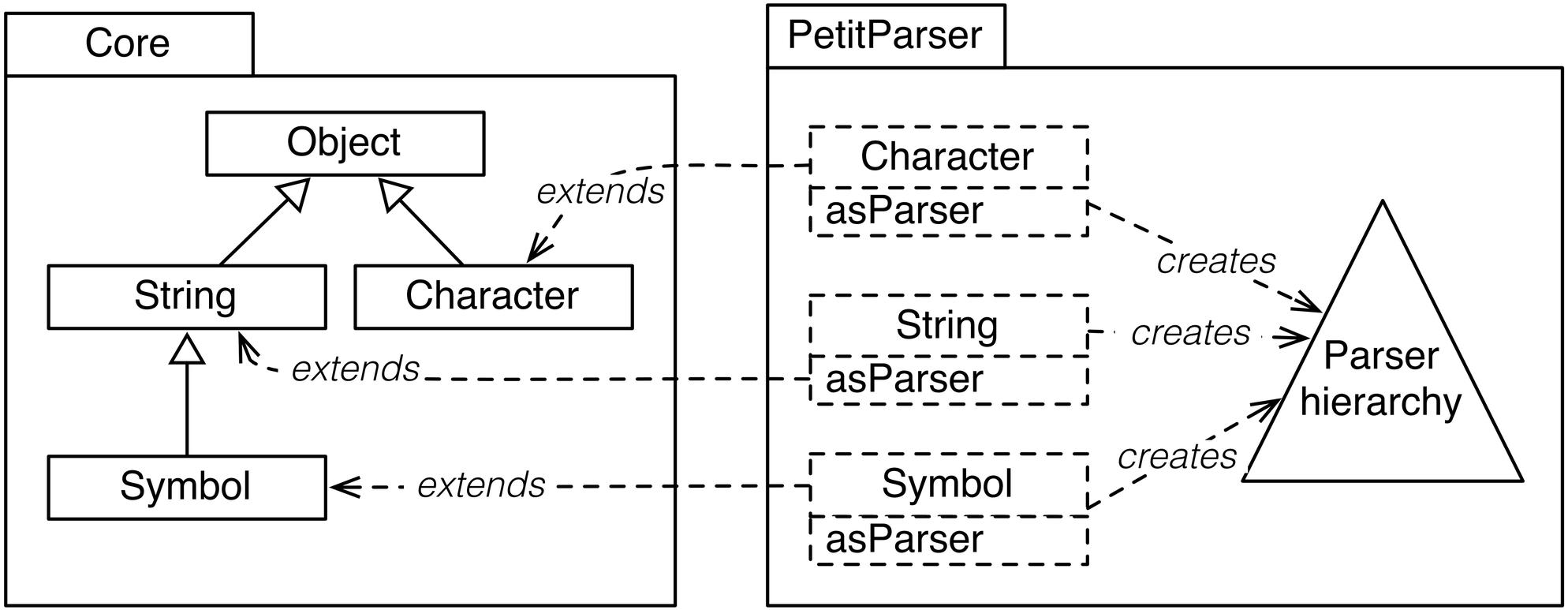
Extensions Break Modularity

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

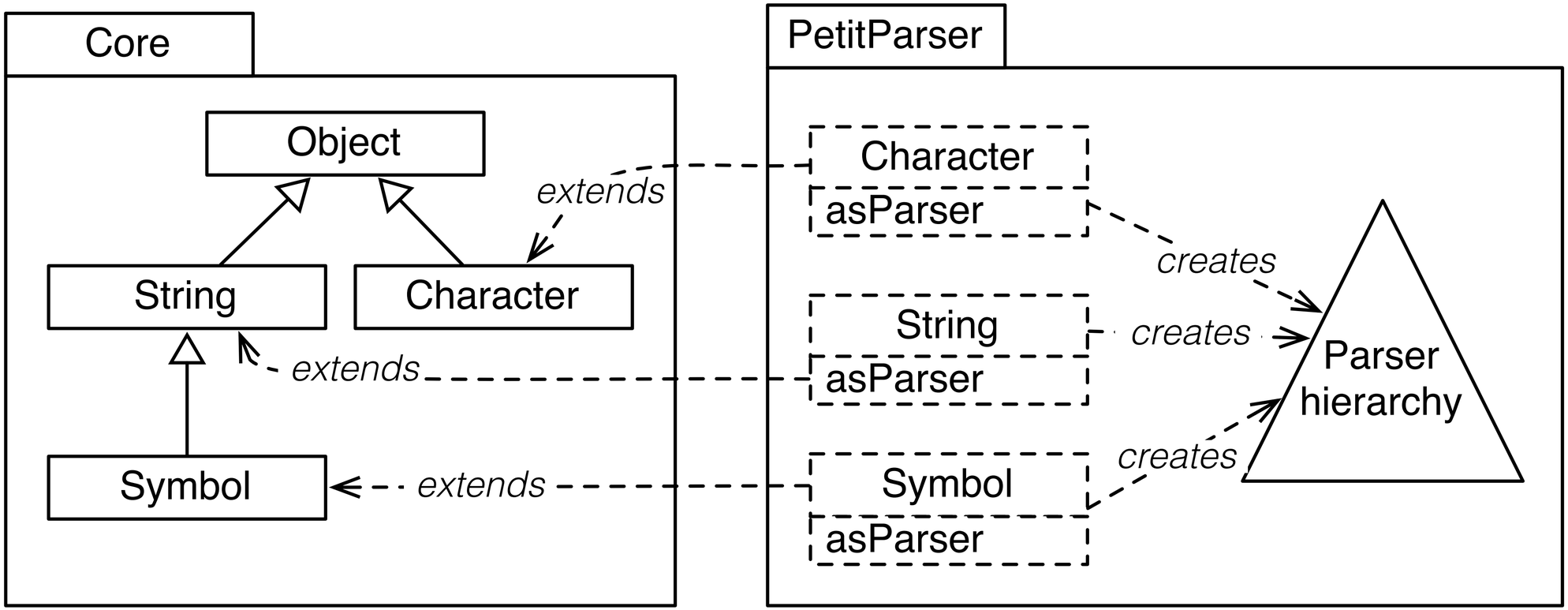
```
Parser >> asParser
  ^ self
```

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

- Monkey patching

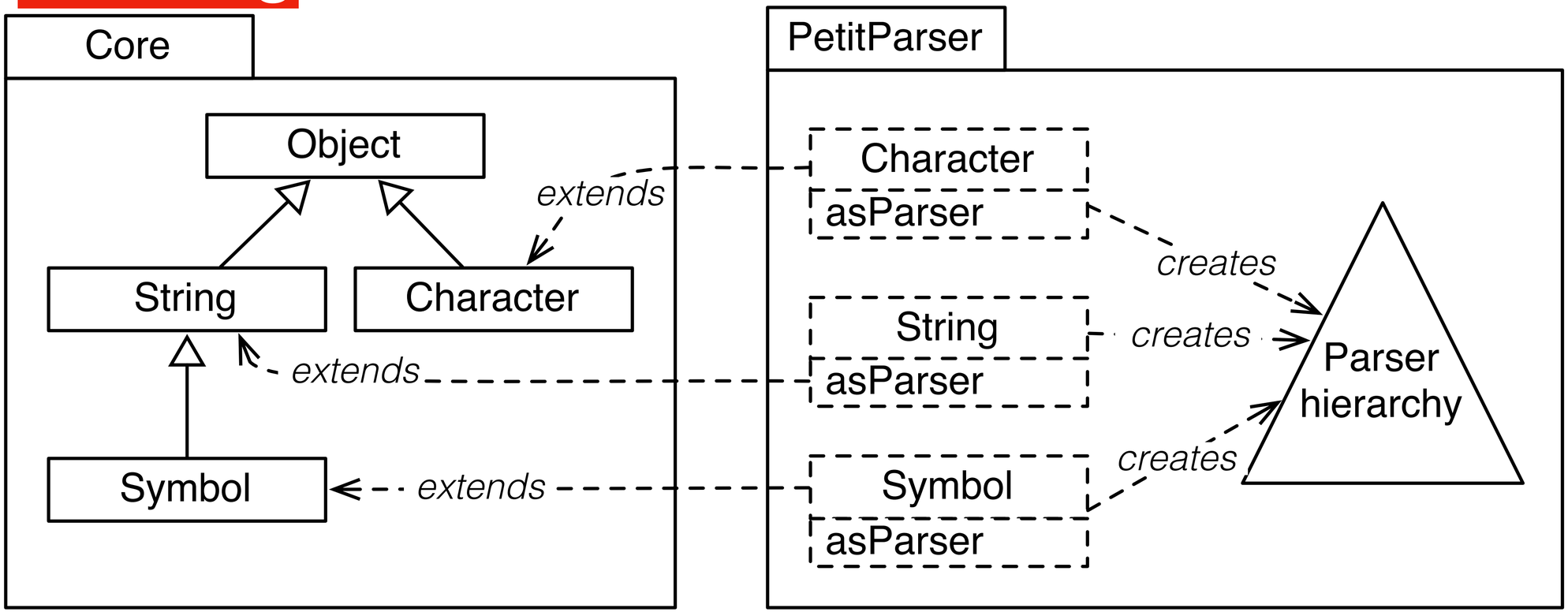
Extension Methods

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



Why Extension Methods are EVIL

- ~~Meth~~ **Maybe breaking** ~~ing~~ classes that **belong to ~~other~~ packages** **you don't own**

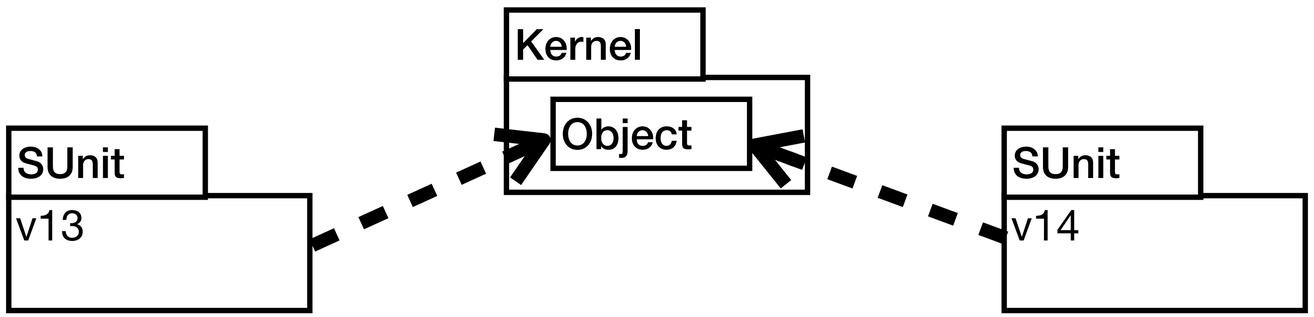


A Vision

Don’t break
my toys,
I’ll not break yours

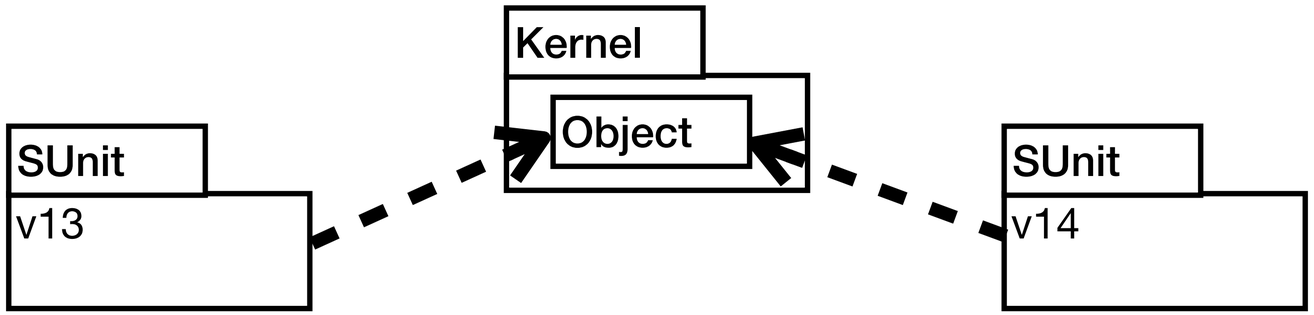
Global Extensions Prevent Multi-Versionning

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



Global Extensions Prevent Multi-Versionning

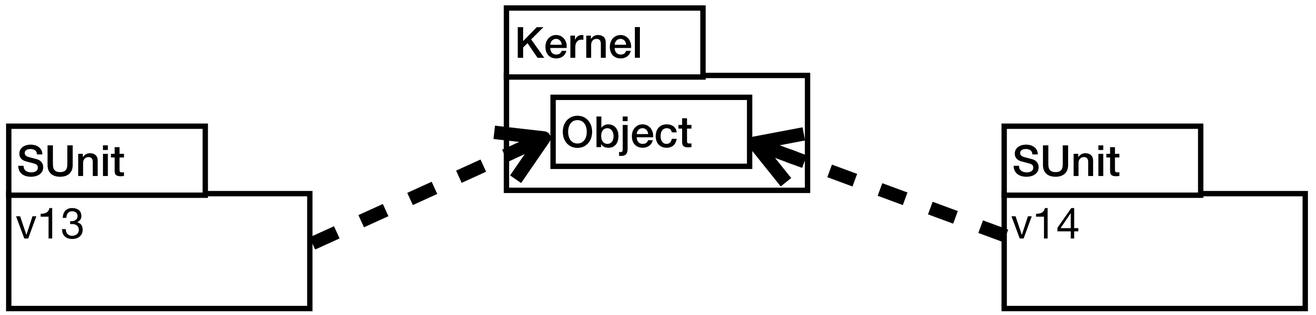
- Two modules extend the same class with the same selector
- Conflict! Who wins?
- Nowadays, last one wins
- And **breaks** the first one



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

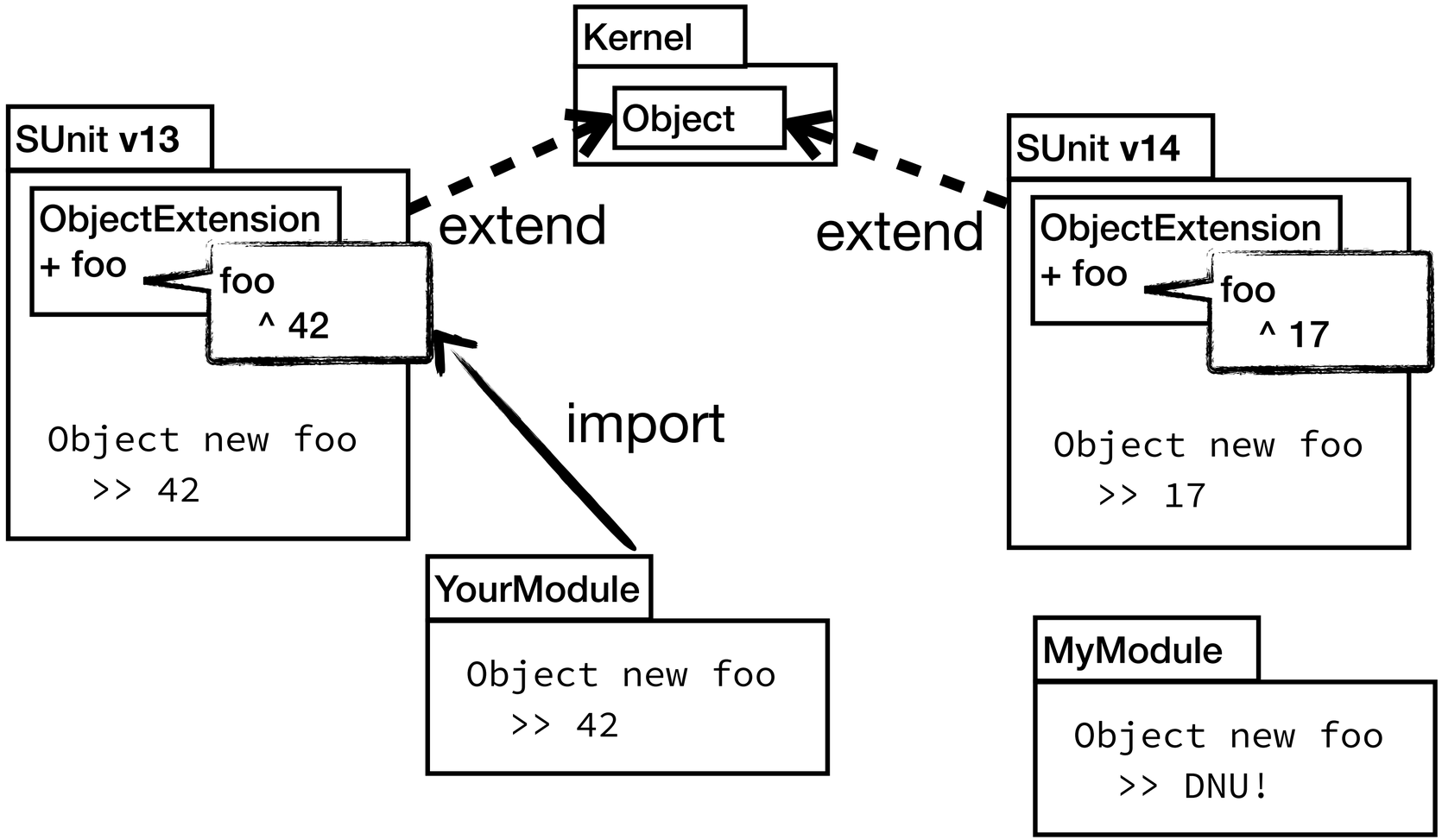


Controlling Extension Methods

- 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

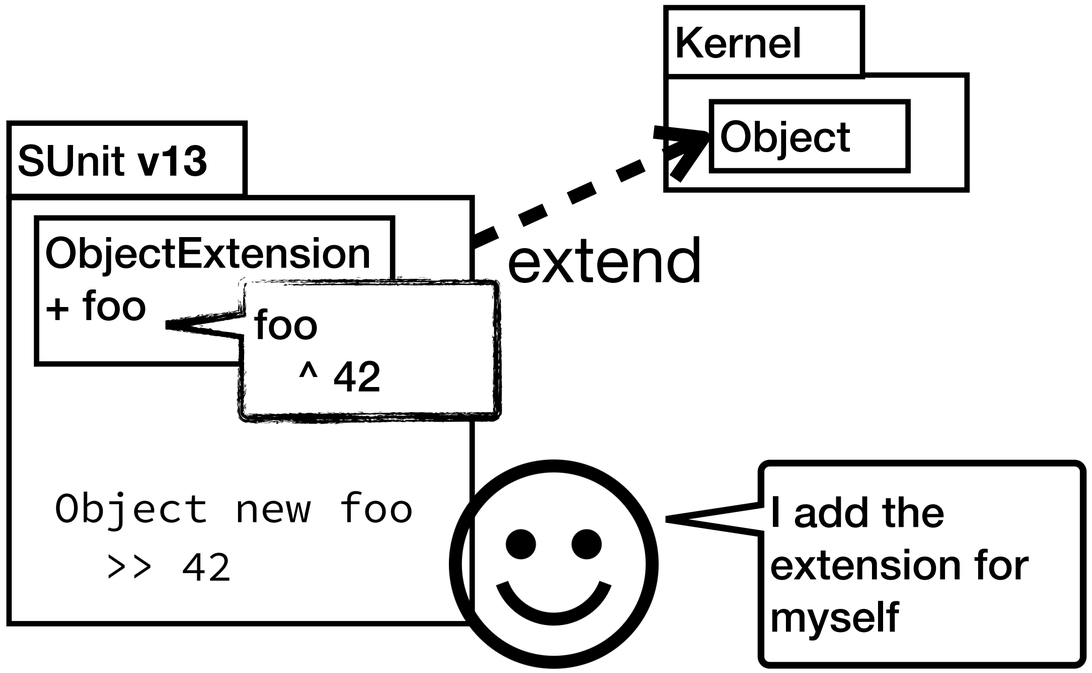


Sneak Peek at Extensions

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

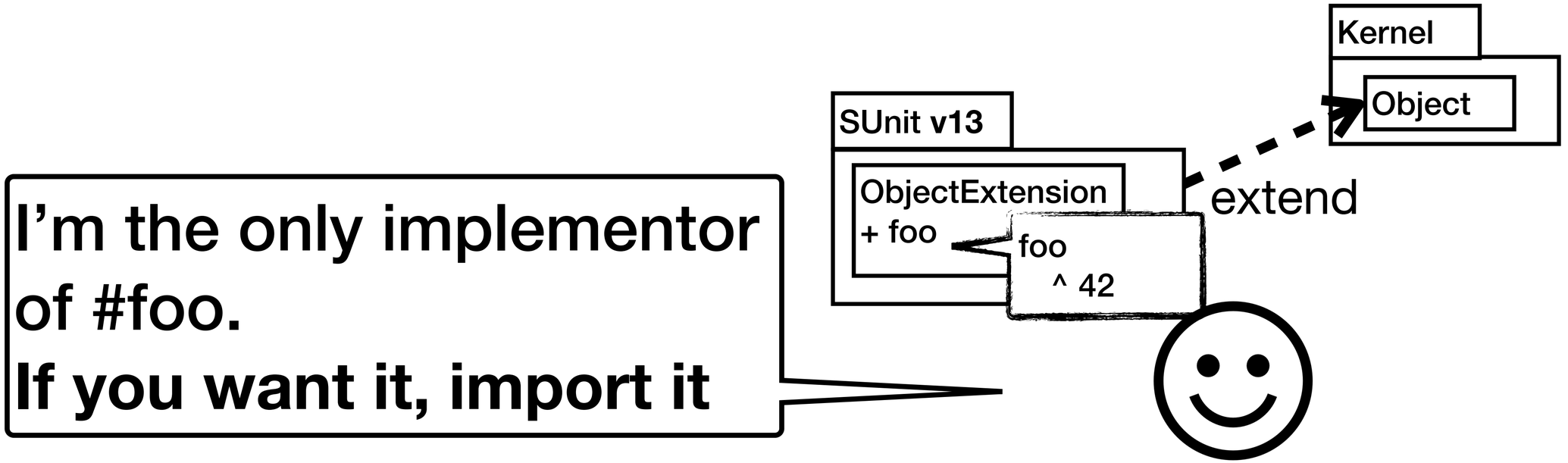
Lots of Extensions are *Local*

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

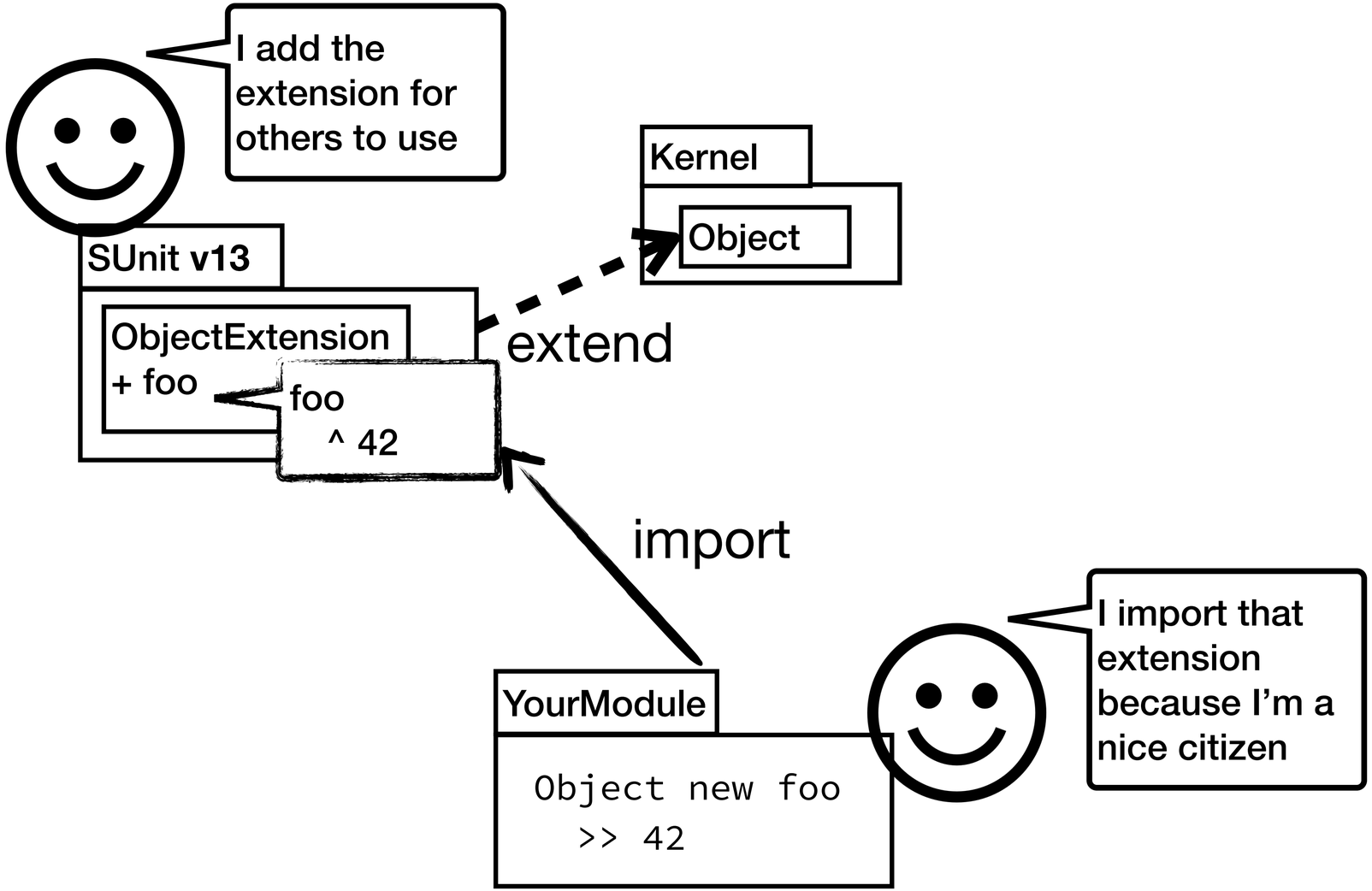


Lots of Extensions are *Monomorphic*

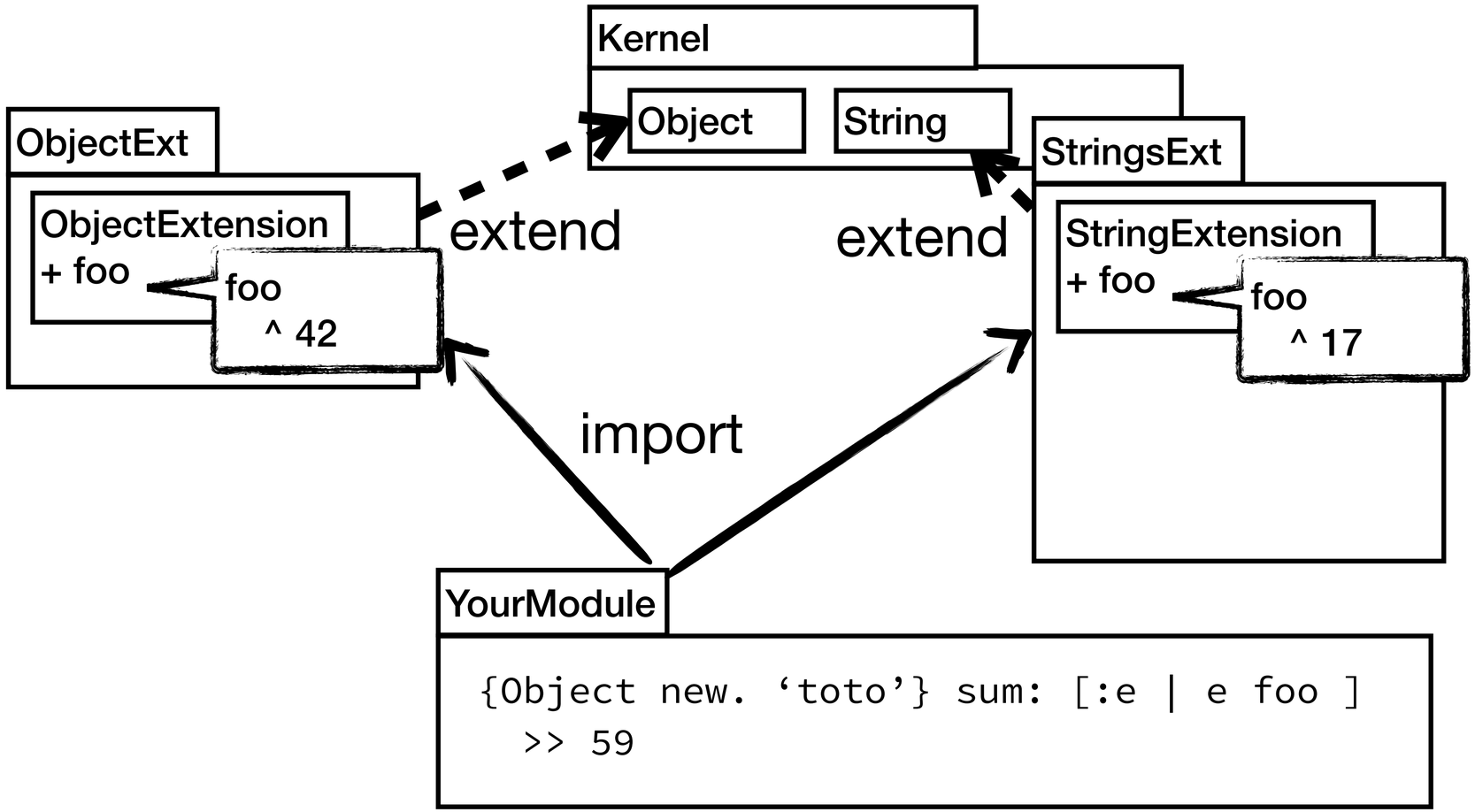
- 1645 (~1/3) are *static* ***monomorphic extensions***
- Unique implementor, no polymorphism



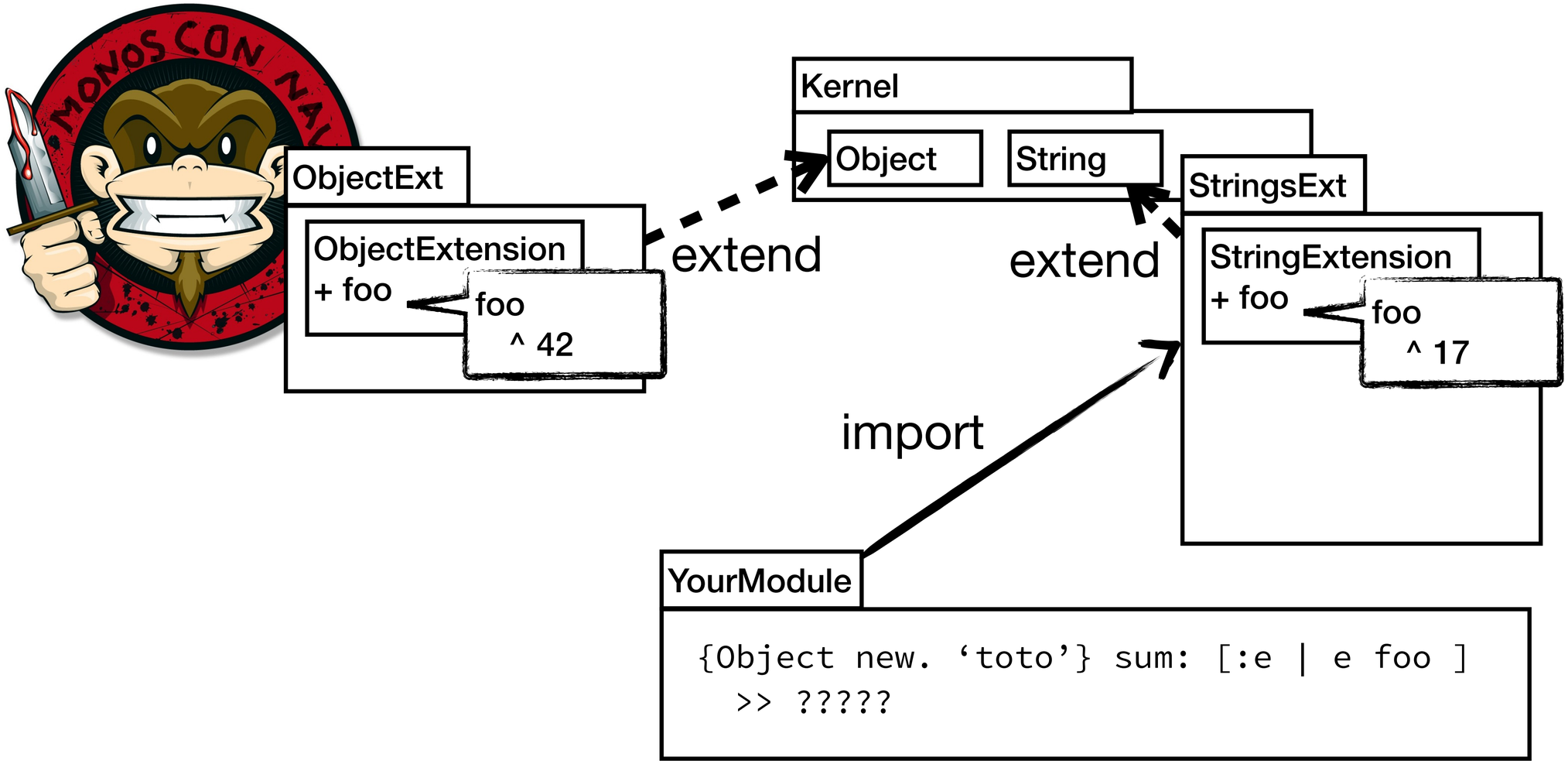
Lots of Other Good Extension Users



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

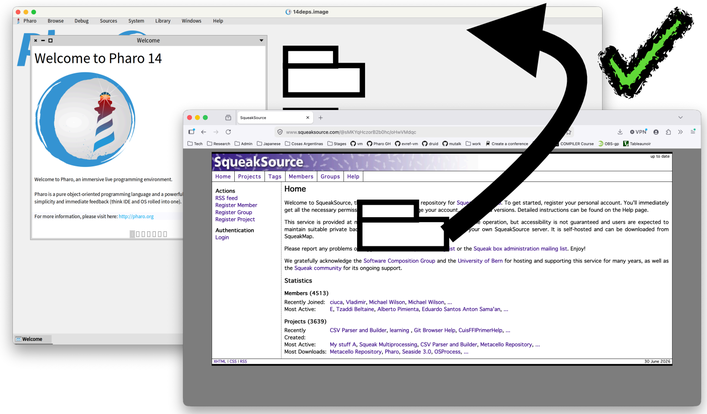
What’s Done, What’s Next?

- Module system + Separate compilation 
- PATH based import system 
- Scoped extensions with a coloring system 

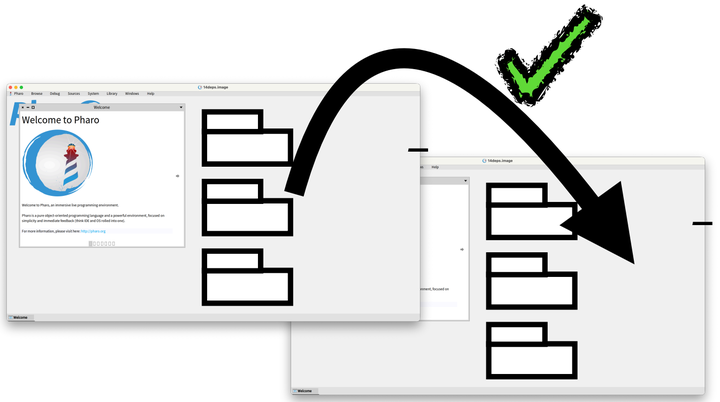
- Module-aware Tools 
- Reflective access to classes! Lots of libraries assume single namespace :(
- Scoped extensions optimizations 

Modules = Namespaces ++

- = Name spaces + multi version + compatibility layers
- + scoped extensions



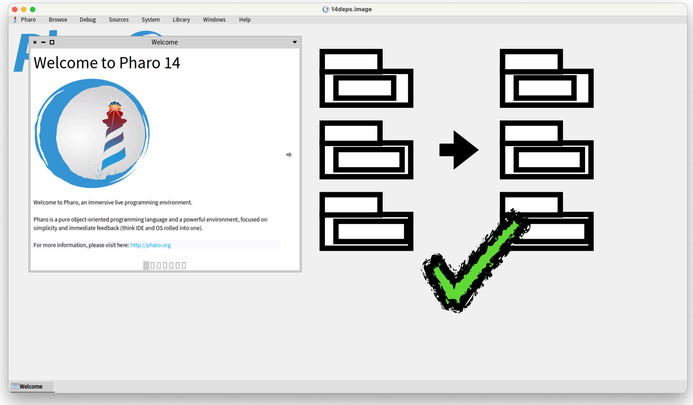
- As much **backward** compatible as possible



- Extra thanks to Luc, Stef, Nathan!

guillermo.polito@inria.fr
@guillep

Université de Lille



Inria

Evref

cnrs

CRIStAL
Centre de Recherche en Informatique, Signal et Automatique de Lille

Scoped Extension Methods