Classboxes: A Minimal Module Model Supporting Local Class Extension

Alexandre Bergel, Stéphane Ducasse,
and Roel Wuyts

Software Composition Group

University of Bern (Switzerland)

bergel@iam.unibe.ch

Outline

1. Class extension and Package
2. Supporting Unanticipated Changes
3. The Classbox Model
4. Implementation
5. Conclusion
6. Future Work

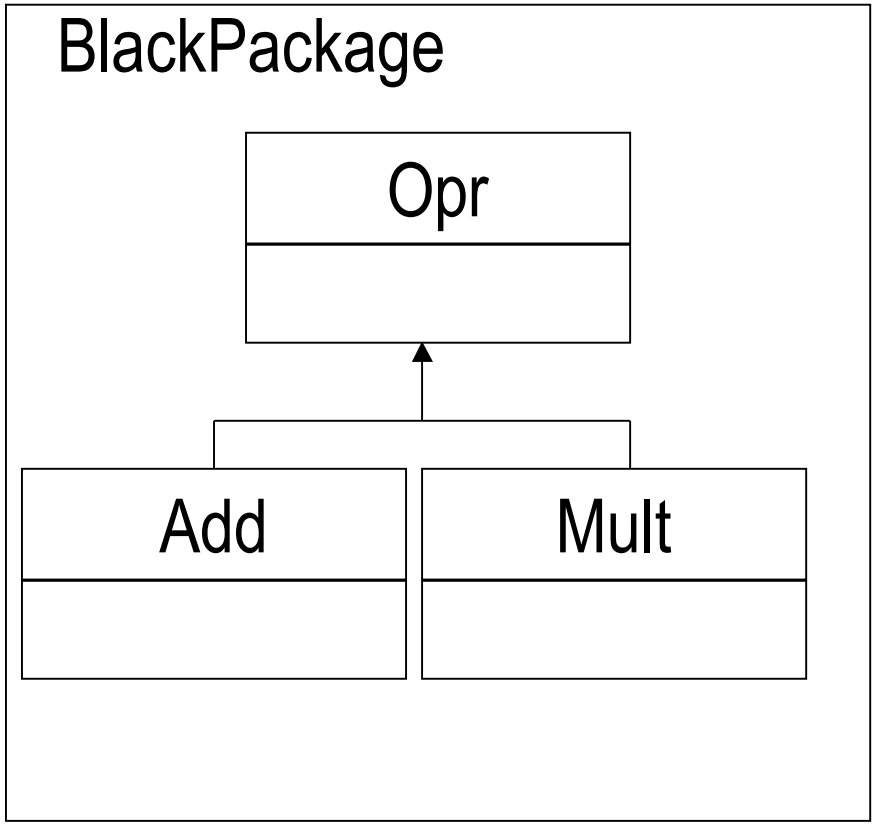
Class Extension

- Languages such as CLOS or Smalltalk provide a mechanism of *Class Extension*

- A *Class Extension* is the ability to **add** or **redefine** methods on a class after its creation

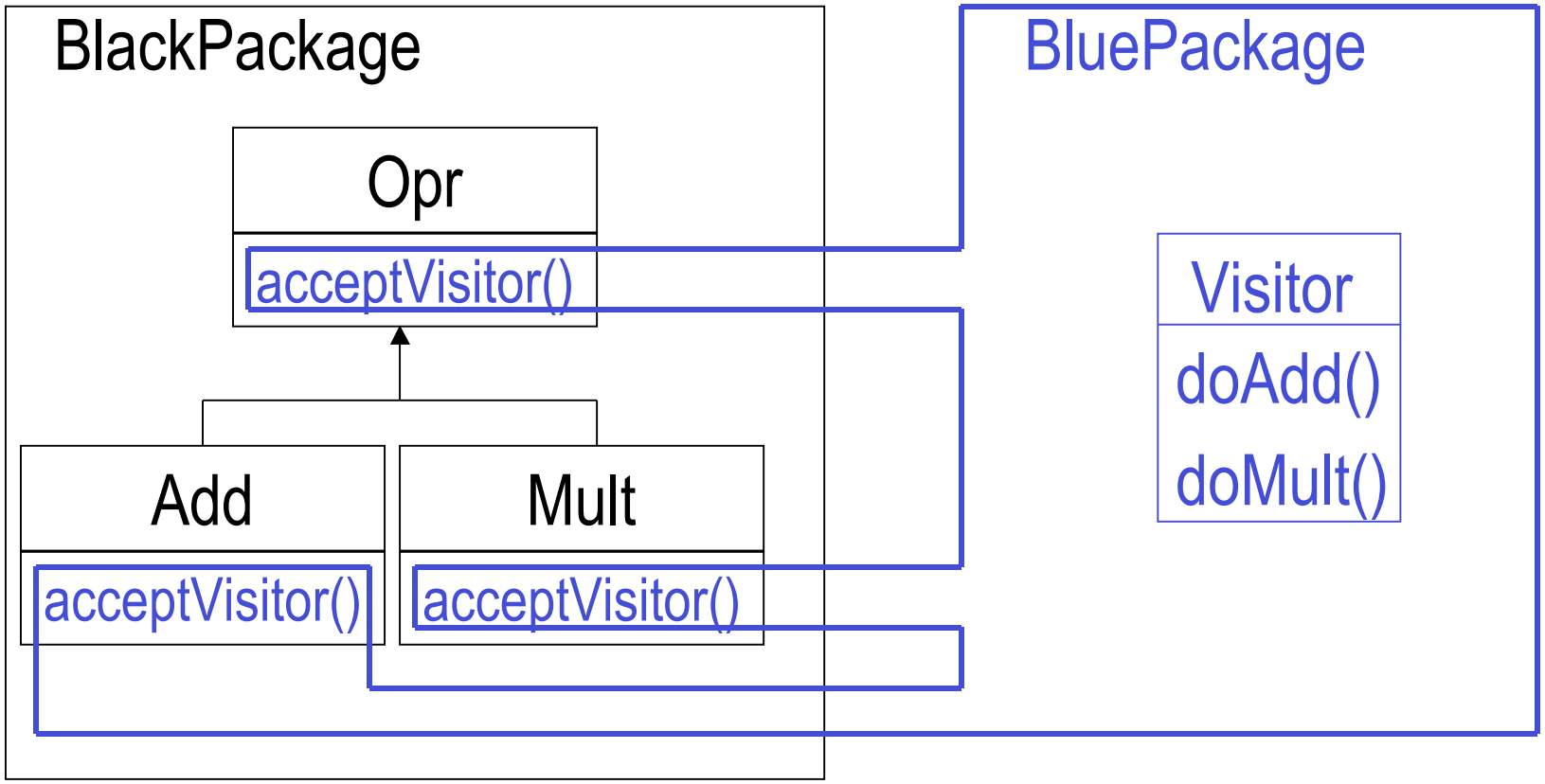
Class Extension: A powerful Mechanism

Visitor Example:

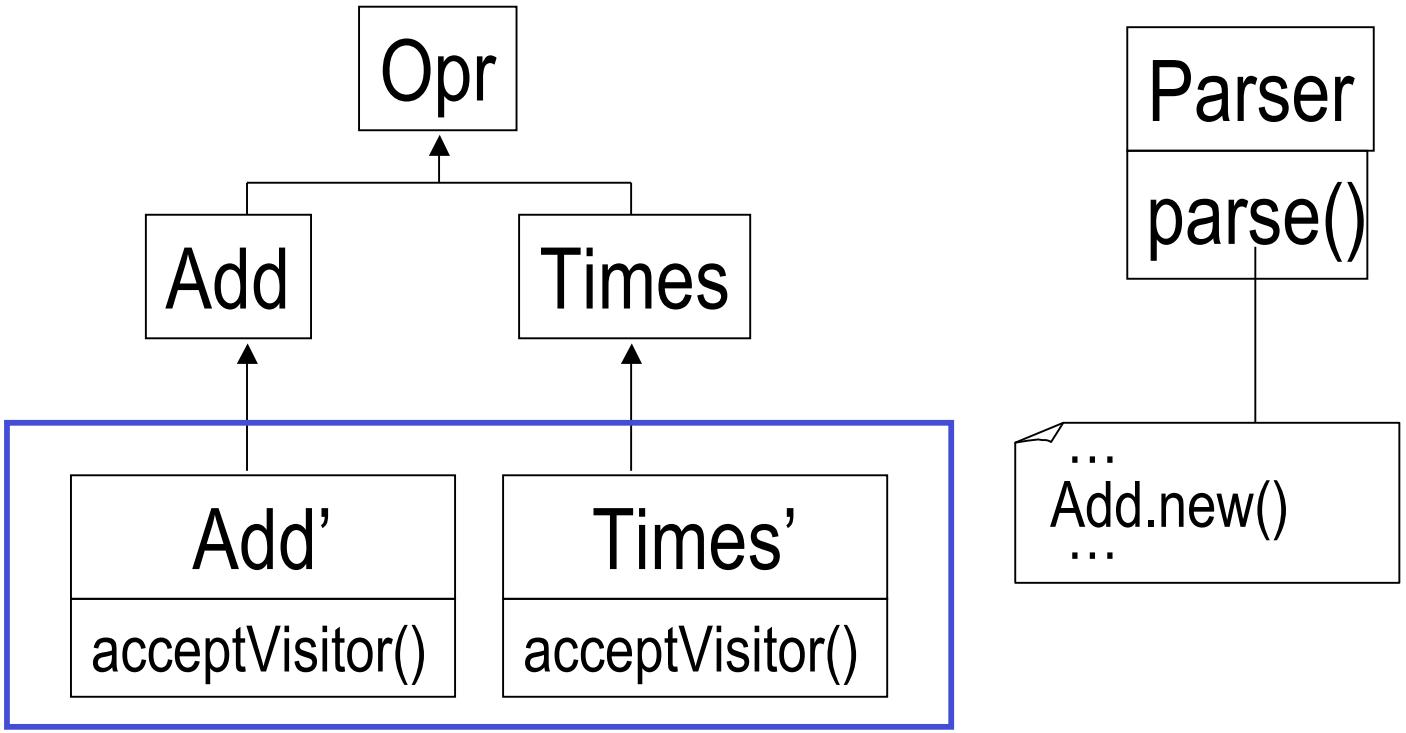


Class Extension: A powerful Mechanism

Visitor Example:



Subclassing does not Solve it



Subclasses are not referenced by the tree constructor

In Smalltalk

- Class extensions are global
- Any application can modify any class in the system

Consequences

- Conflicts may arise (e.g., two applications may add the same method)
- Robustness aspect (e.g., an application may redefine a critical method)

Supporting Unanticipated Changes: Adaptation

- Java, Modula-3,…have *package/modules* but no Class Extension
- Smalltalk has *Class Extension* but no package/module

  Class Extension + Module ?

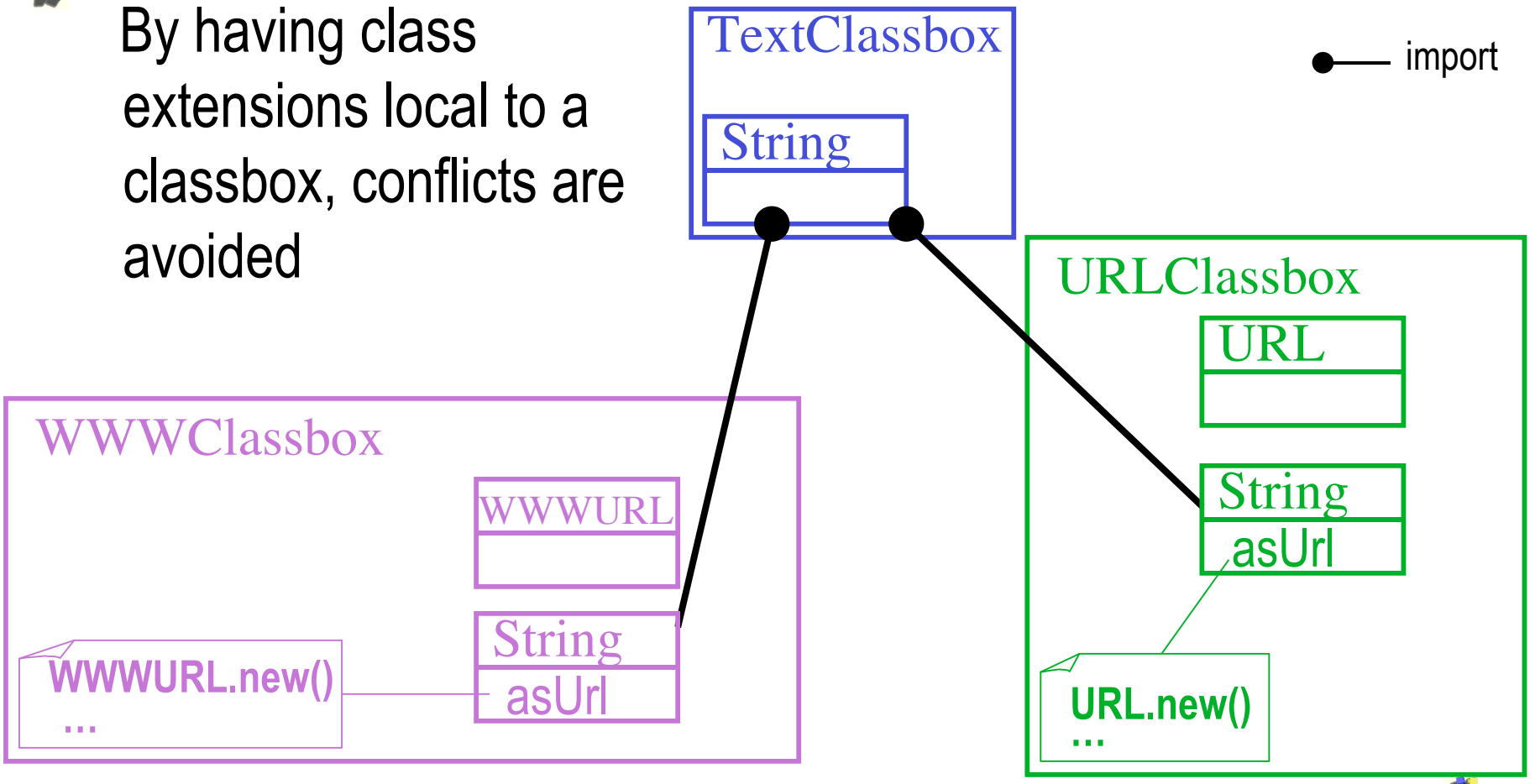
- How to control class extensions?
- How to apply a set of unanticipated changes without breaking existing clients?

Classbox Model Definition

- The Classbox Model is a minimal module model supporting local class extension
- A Classbox
  - Is a unit of *scoping* (i.e., it acts as a namespace)
  - Can define *classes*
  - Can *import* class definitions from other classboxes
  - Can define *methods* on classes in the scope: class extensions
- Classes and methods always belong to exactly one classbox
- Visibility of each element in a classbox is bound to this classbox

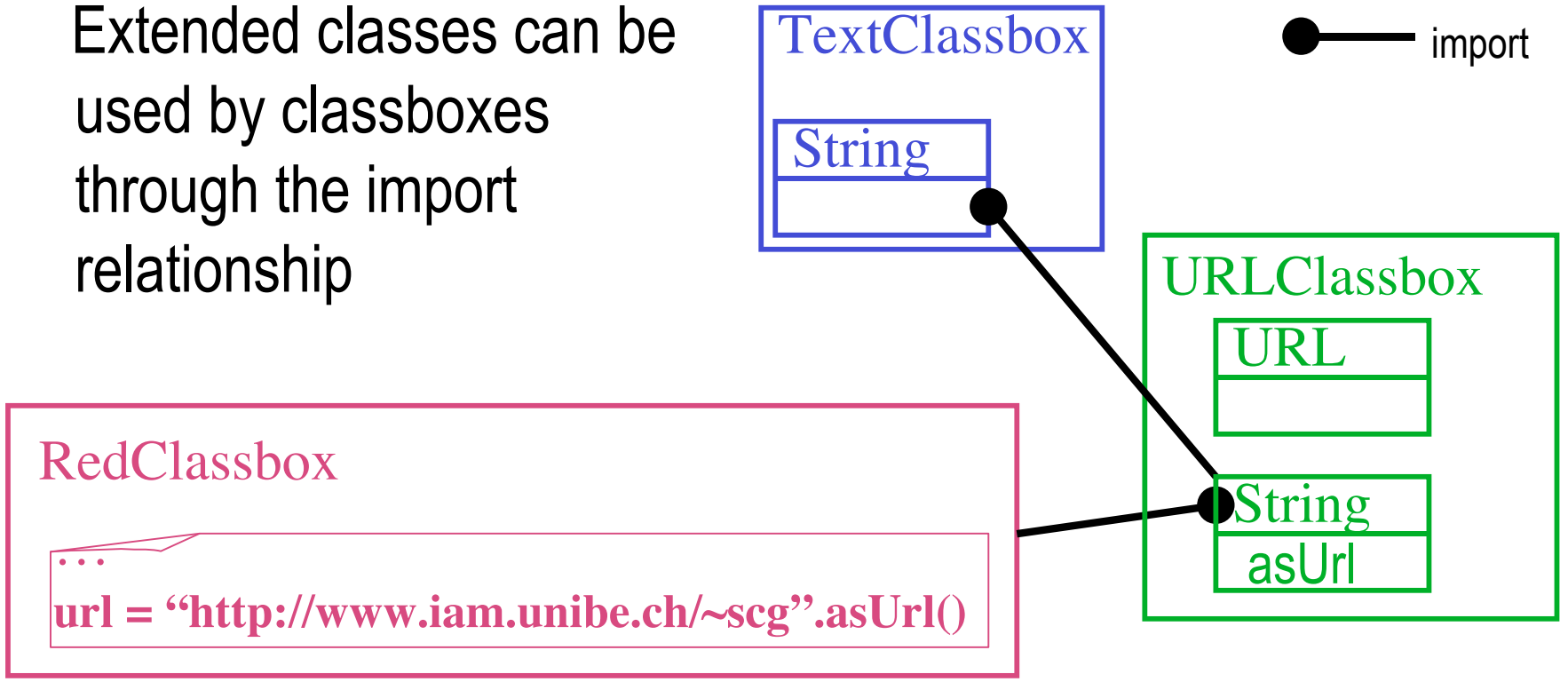
Classbox Model: Avoiding Conflict

By having class extensions local to a classbox, conflicts are avoided



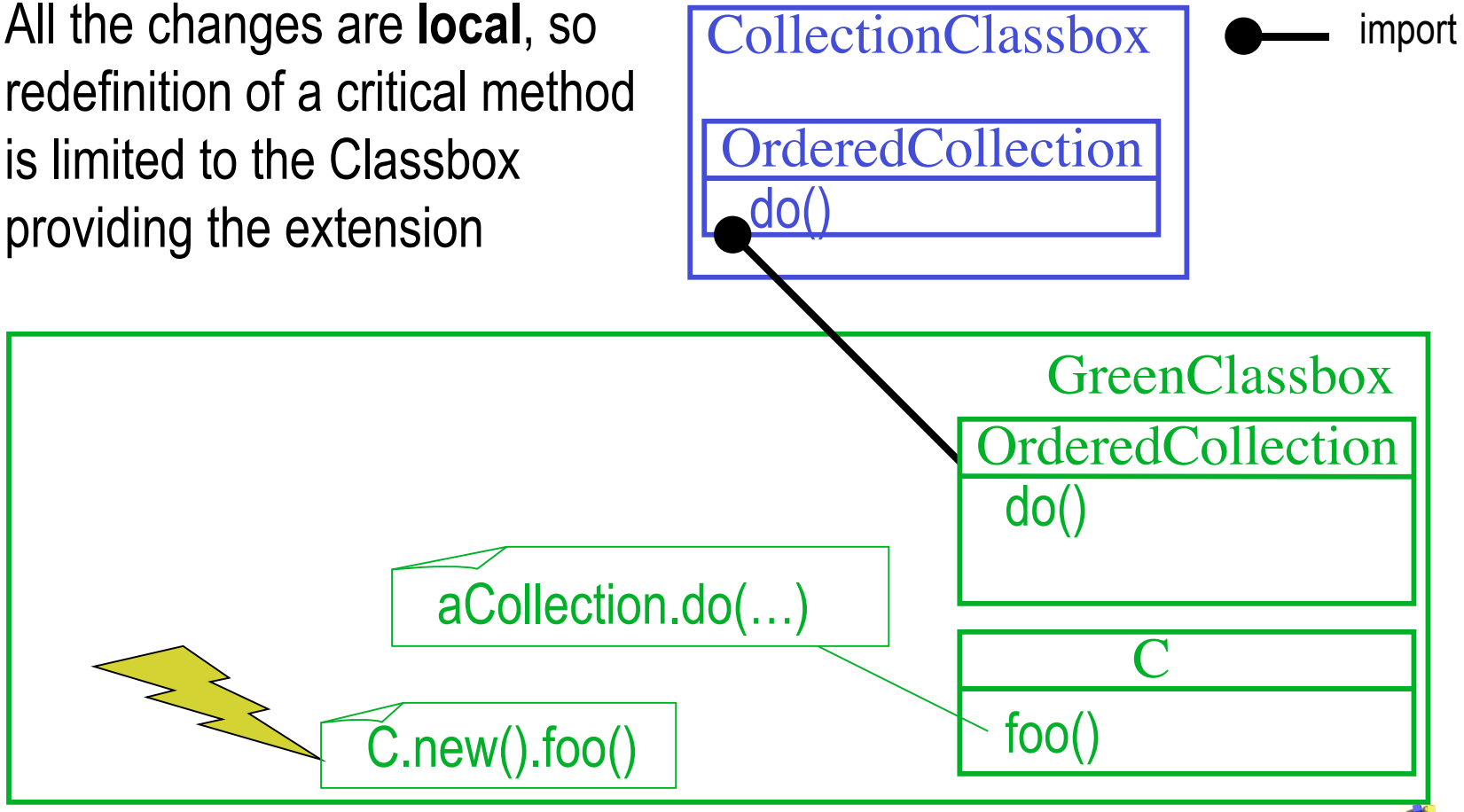
Classbox Model: Import and Visibility

Extended classes can be used by classboxes through the import relationship



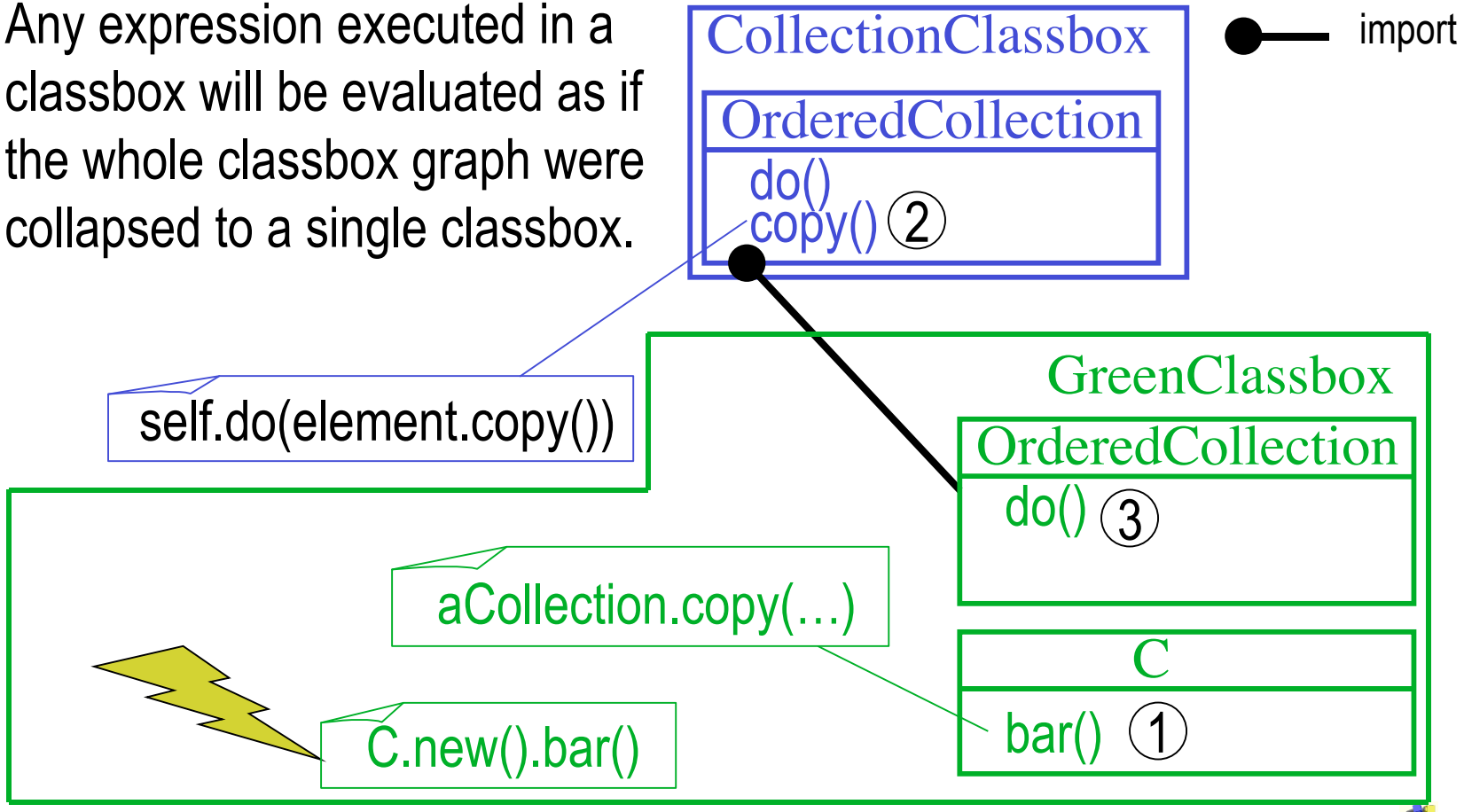
Classbox Model: Classbox as a Sandbox

All the changes are **local**, so redefinition of a critical method is limited to the Classbox providing the extension

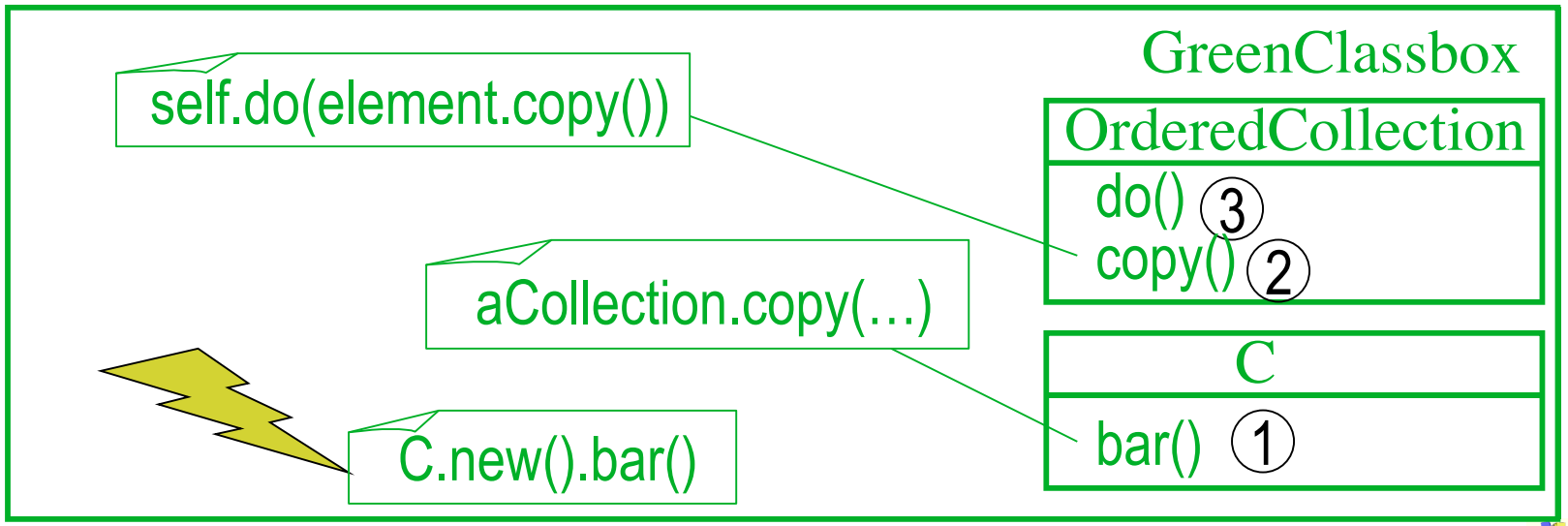


Classbox Model: Flattening Property

Any expression executed in a classbox will be evaluated as if the whole classbox graph were collapsed to a single classbox.



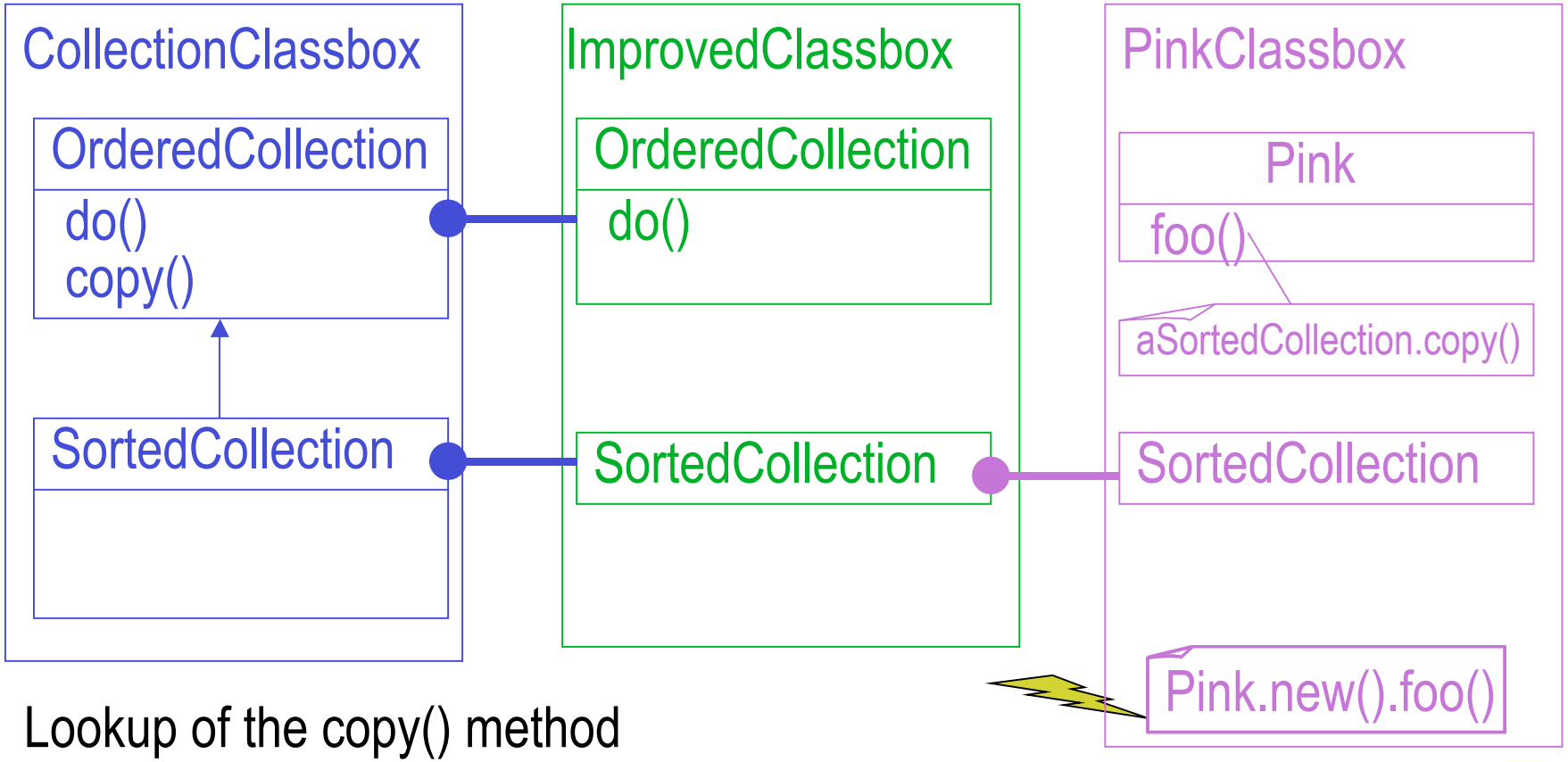
From within the Green Classbox



Implementation

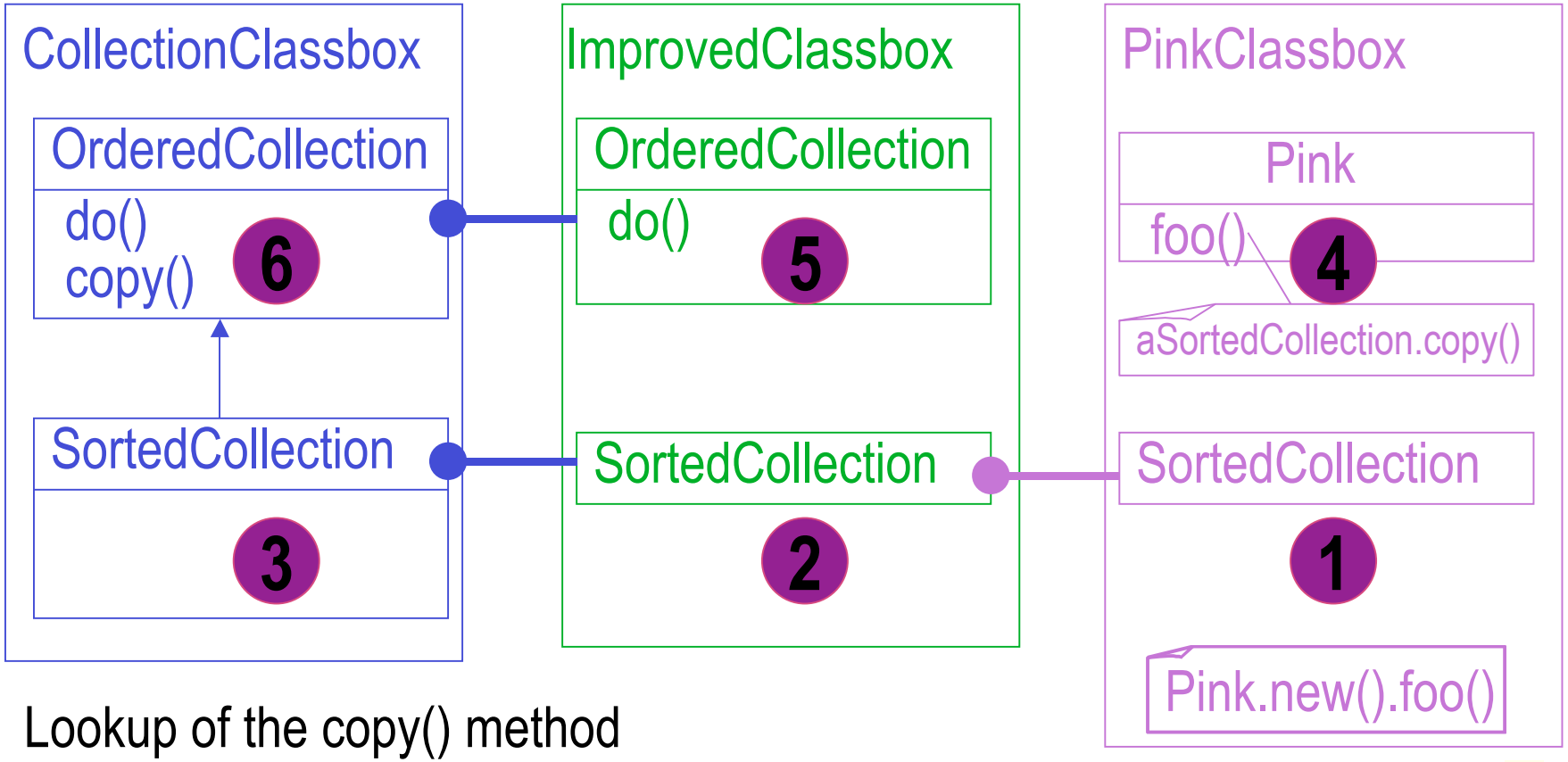
- Implemented in Squeak, an open-source Smalltalk
- To avoid excessive memory costs:
  - modify the VM
- New method lookup taking some extra parameters into account such as:
  - Original classbox (referring the classbox where the initial expression is executed)
  - Collection of all the traversed classboxes

Lookup: Imports before Inheritance



Lookup of the copy() method

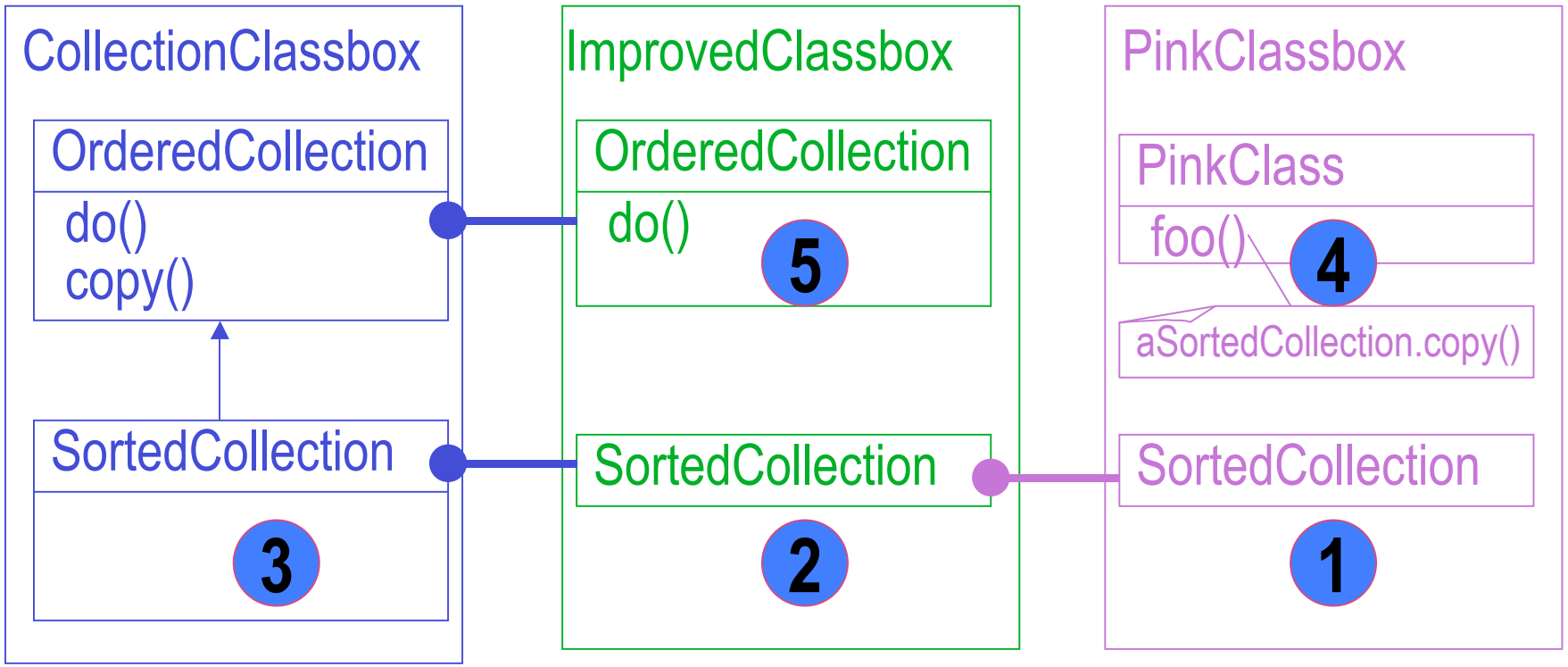
Lookup: Imports before Inheritance



Lookup of the copy() method

Lookup: Imports before Inheritance

The copy() method invokes do() one



Without any drastic optimization, a system is 60% slower

Conclusion

- Definition of a module system for controlling class extensions:
  - Method addition and replacement are only visible in the module that defines them
  - Control the visibility
  - Support for unanticipated evolution
- New method lookup
  - This has a price: slow performance with the current implementation

Future Work

- Allow *aliasing,* when importing
- Refined visibility control (allowing an extension to be global: evolution of a classbox)
- Add multiple Classbox *interfaces (*clients may see a classbox under different views*)*
- *Formalization* to specify the flattening property

Contact Information

- Alexandre Bergel (bergel@iam.unibe.ch)
- Website:
  - http://www.iam.unibe.ch/~scg/Research/Classboxes/index.html