RMod

Inria

Université de Lille

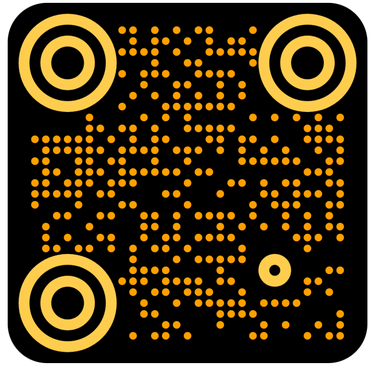
cnrs

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

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

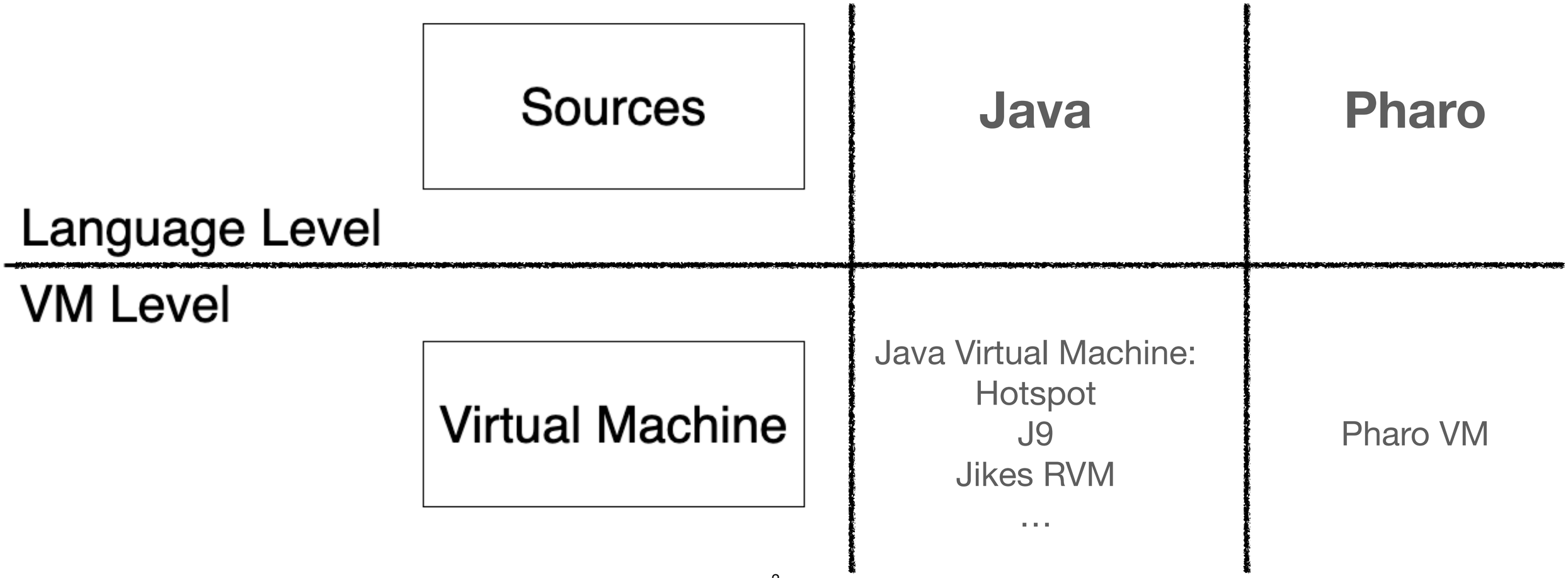
Ease Virtual Machine Level Tooling With Language Level Ordinary Object Pointers

Pierre Misse-Chanabier
Theo Rogliano



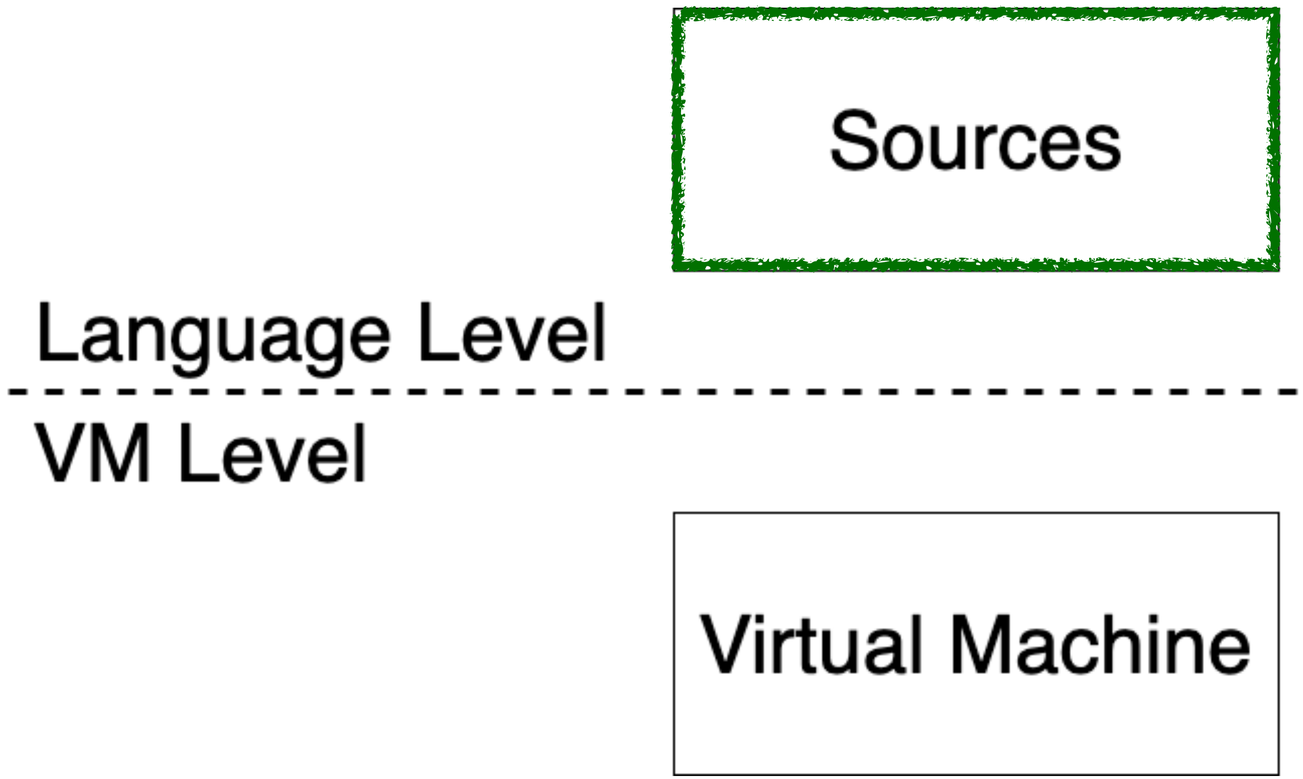
Who Does Not Love Tools ?

Tooling Levels



Who Does Not Love Tools ?

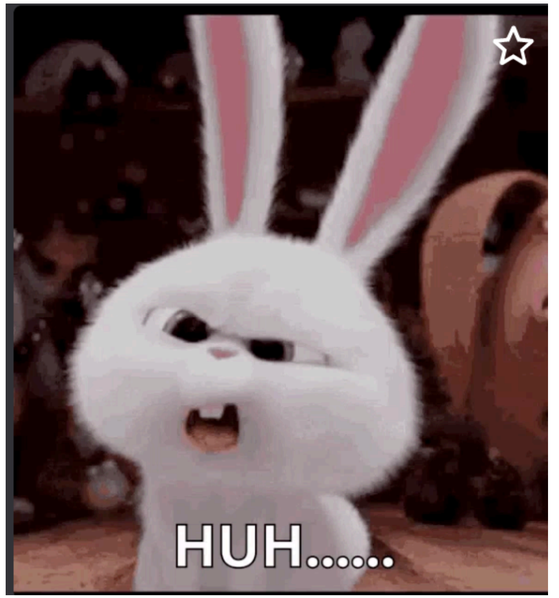
Tools at the Language Level



Debuggers, Profilers, Compilers, Code Browser,
XUnit, Refactoring, Project Management
… … …

Who Does Not Love Tools ?

Tools at the VM Level



Sources

Language Level

VM Level

Virtual Machine

Debuggers, Profilers, Memory visualisation, Bootstrap
Difficult to write, requires expertise

Who Does Not Love Tools ?

Why Should we Care About VM Level Tools ?

```
Form >> #scaledByDisplayScaleFactor
        self halt.
        ^ self scaledToSize: self extent * self currentWorld displayScaleFactor
```

Who Does Not Love Tools ?

Don’t Close the Environment !

```
Form >> #scaledByDisplayScaleFactor
        self halt.
        ^ self scaledToSize: self extent * self currentWorld displayScaleFactor
```

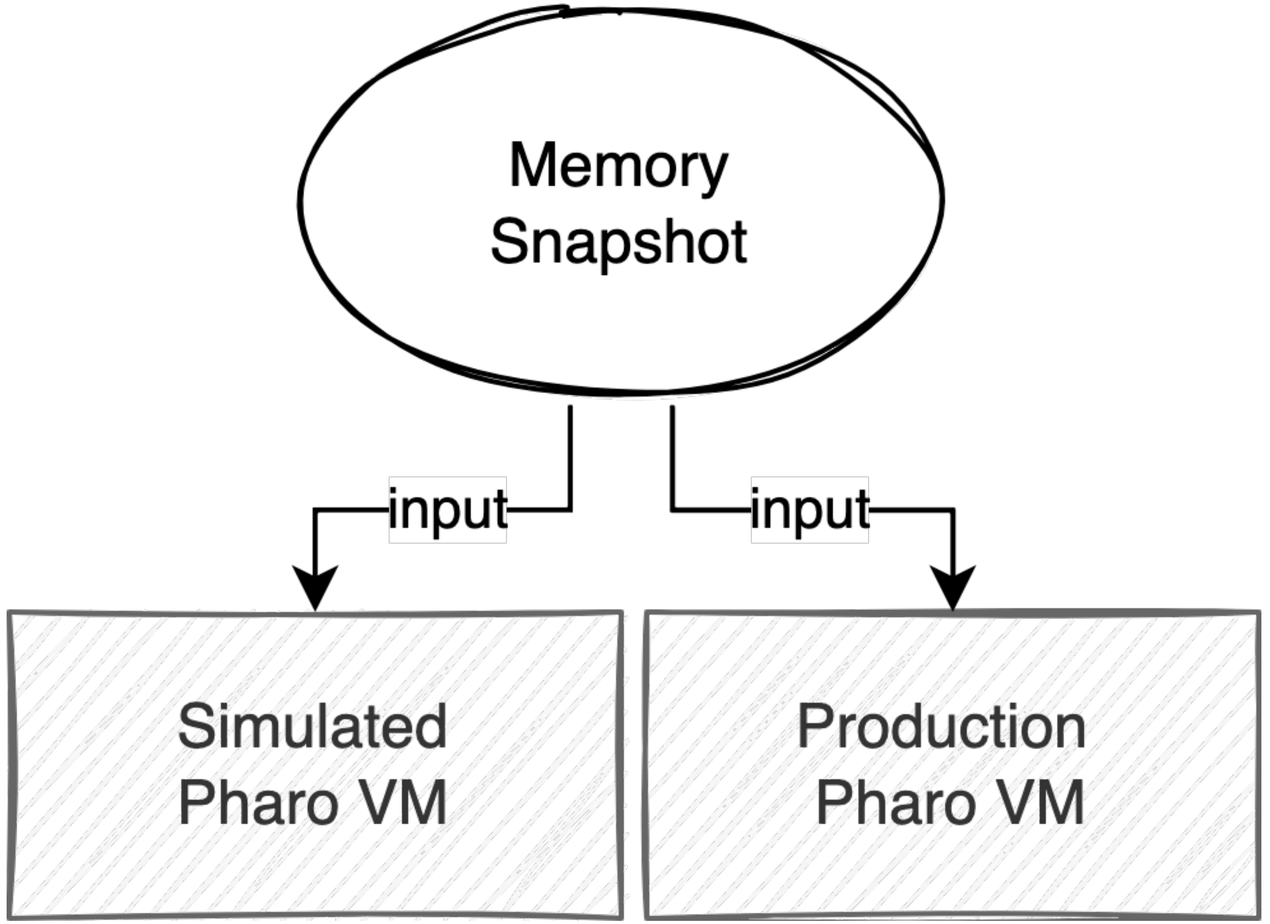
Who Does Not Love Tools ?

We need the VM support !

```
Halt
SmallInteger(Object)>>haltOnce
Form>>scaledByDisplayScaleFactor
ThemeIcons>>iconNamed:
MorphicRootRenderer(Object)>>iconNamed:
MorphicRootRenderer(OSWorldRenderer)>>setAttributesDefault
MorphicRootRenderer class(OSWorldRenderer class)>>forWorld:
[ :arg5 | tmp2 := arg5 forWorld: arg1 ] in AbstractWorldRenderer
FullBlockClosure(BlockClosure)>>cull:
[ :arg4 | (arg1 value: arg4) ifTrue: [ ^ arg2 cull: arg4 ] ] in
 arg2 cull...etc...
OrderedCollection>>do:
OrderedCollection(Collection)>>detect:ifFound:ifNone:
OrderedCollection(Collection)>>detect:ifFound:
AbstractWorldRenderer class>>detectCorrectOneForWorld:
```

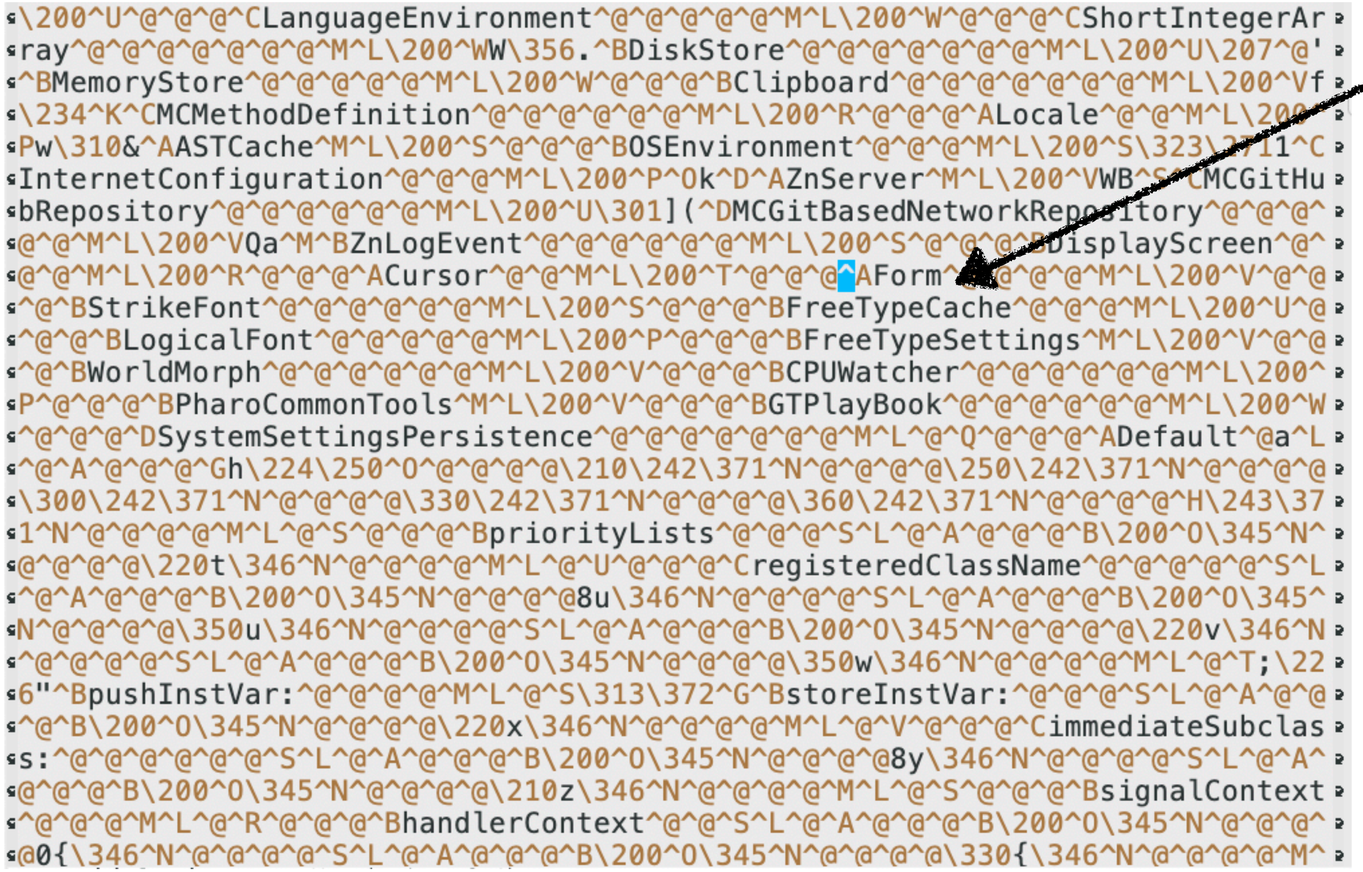
Let’s Code VM Level Tools !

Looking for the Class Form in Memory …



Let’s Code VM Level Tools !

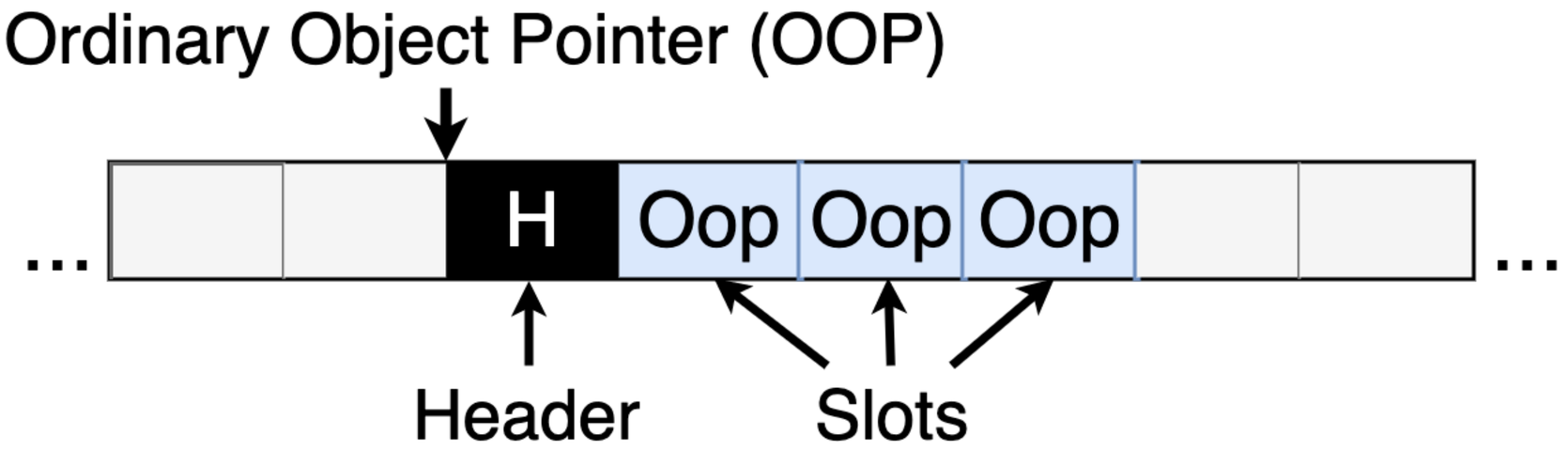
… With the Help of the VM Simulator



Is this it ?

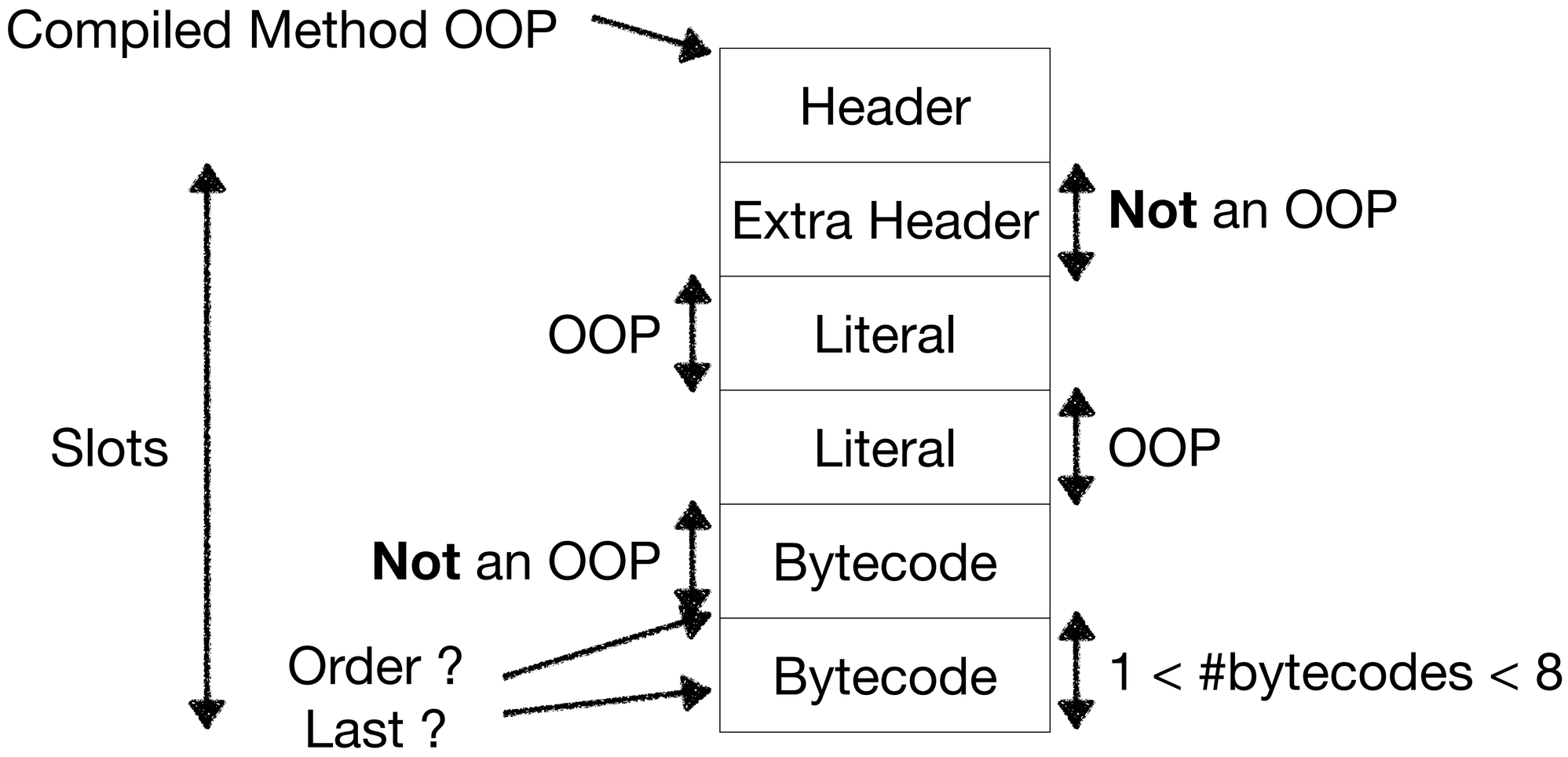
Let’s Code VM Level Tools !

Ordinary Object Pointer (OOP)



Let’s Code VM Level Tools !

Method Layouts



Let’s Code VM Level Tools !

Finding a Class at the VM Level

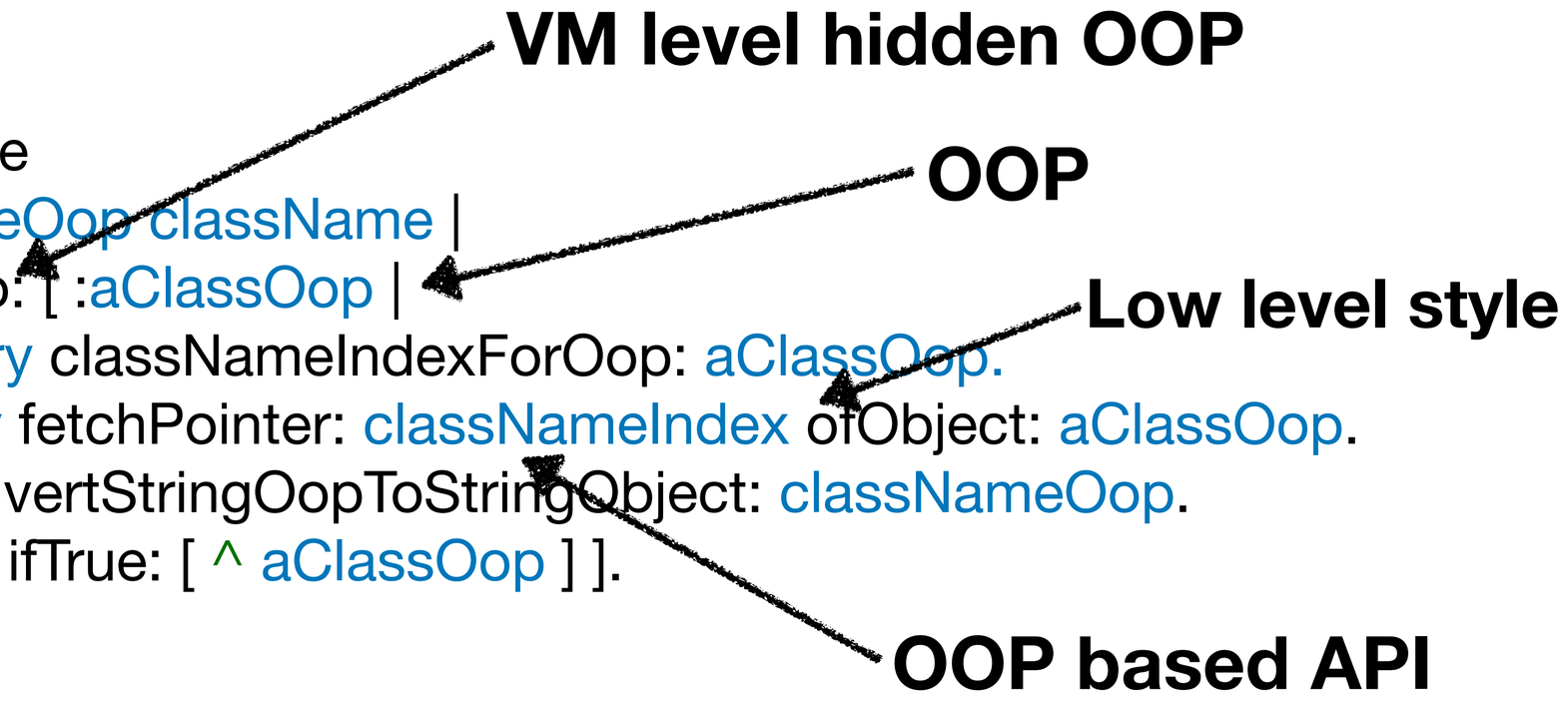
```
findClassNamed: aClassName
    | classNameIndex classNameOop className |
    memory classTableEntriesDo: [ :aClassOop |
        classNameIndex := memory classNameIndexForOop: aClassOop.
        classNameOop := memory fetchPointer: classNameIndex ofObject: aClassOop.
        className := memory convertStringOopToStringObject: classNameOop.
        className = aClassName ifTrue: [ ^ aClassOop ] ].
    ^ memory nilOOP
```

memory findClassNamed: #Form >>> 406749864

Let’s Code VM Level Tools !

Knowledge Gaps

```
findClassNamed: aClassName
    | classNameIndex classNameOop className |
    memory classTableEntriesDo: [ :aClassOop |
        classNameIndex := memory classNameIndexForOop: aClassOop.
        classNameOop := memory fetchPointer: classNameIndex ofObject: aClassOop.
        className := memory convertStringOopToStringObject: classNameOop.
        className = aClassName ifTrue: [ ^ aClassOop ] ].
    ^ memory nilOOP
```



Let’s Code VM Level Tools !

Knowledge Gaps recaps

Issues

- Ordinary Object Pointers (OOP)
- API manipulating OOPs
- VM level information

Polyphemus
Introducing LLOOPs

Language level OOPs

Issues

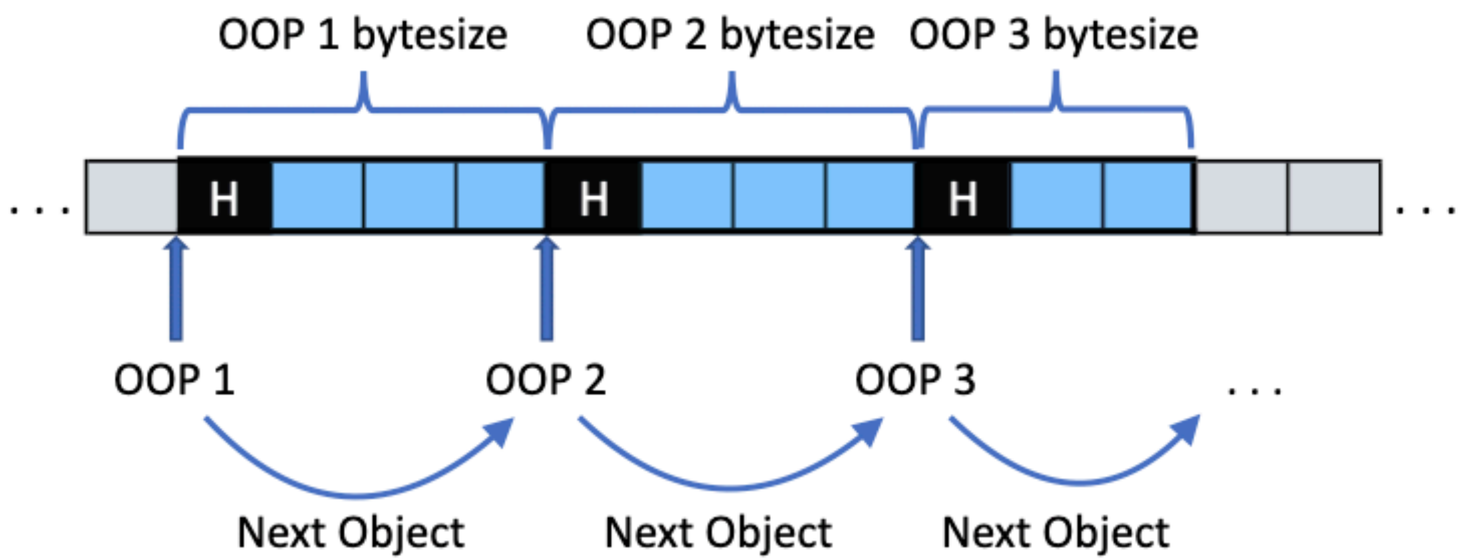
- Ordinary Object Pointers (OOP)
- API manipulating OOPs
- VM level information

Solutions

- Language Level Entities
- Identifying and Typing OOP
- VM and Language level information

Polyphemus

Iterating the Memory



Polyphemus

Identifying the Each OOP

Is it a Class ?

Is it a Metaclass ?

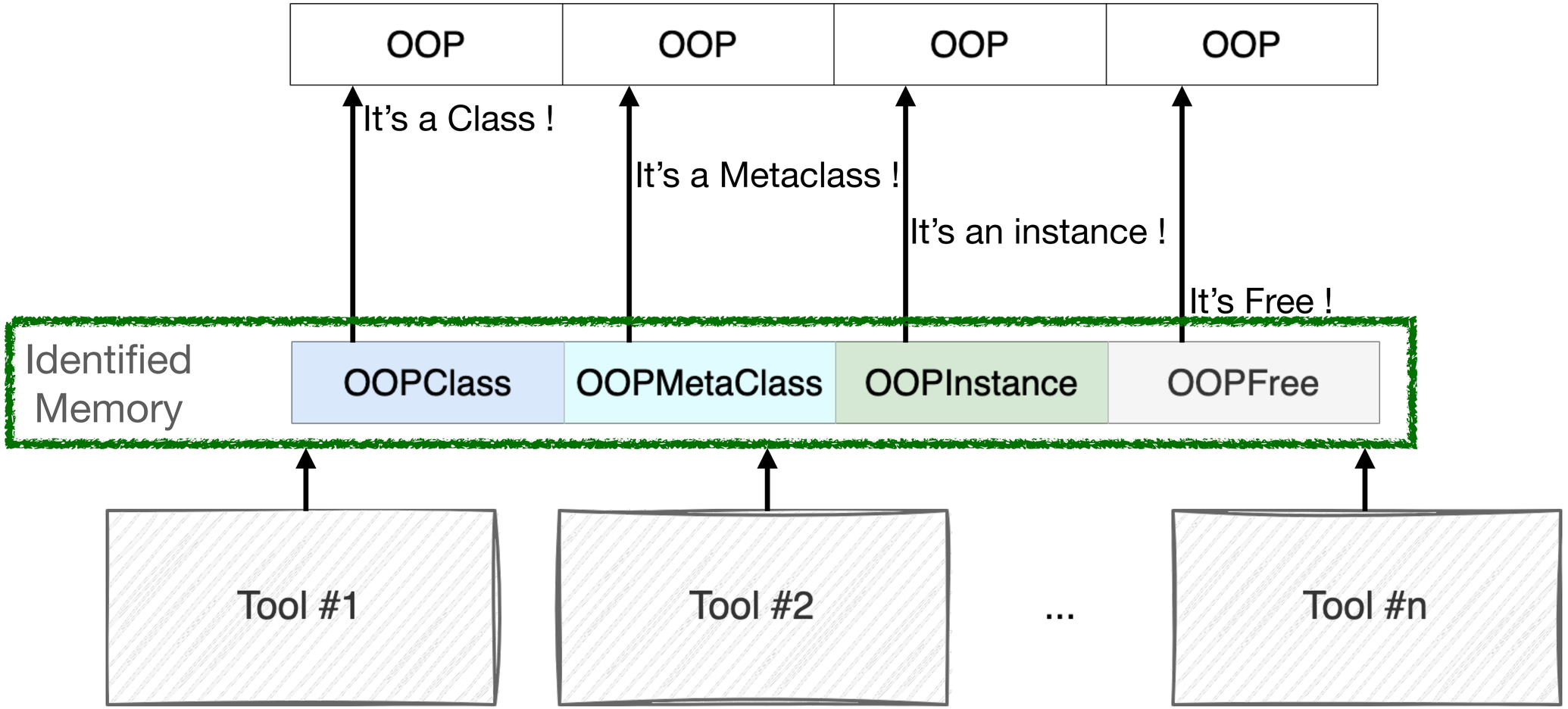
Is it a Free Object ?

Is it a […] ?

If no particularity, it’s an instance !

Polyphemus

LLOOP based tools



Polyphemus Tools
Writing Tools

```
identifiedMemory allClassesOop.
identifiedMemory reifiedMetaclass.
identifiedMemory allClassesOop select: [ :o | o isClassSide ].
identifiedMemory objects
        select: [ :o | o isCompiledMethodOop ]
        thenCollect:[ :aCompiledMethod |
                [ aCompiledMethod decompile ] on: Error do: [ nil ] ].
```

Polyphemus Tools

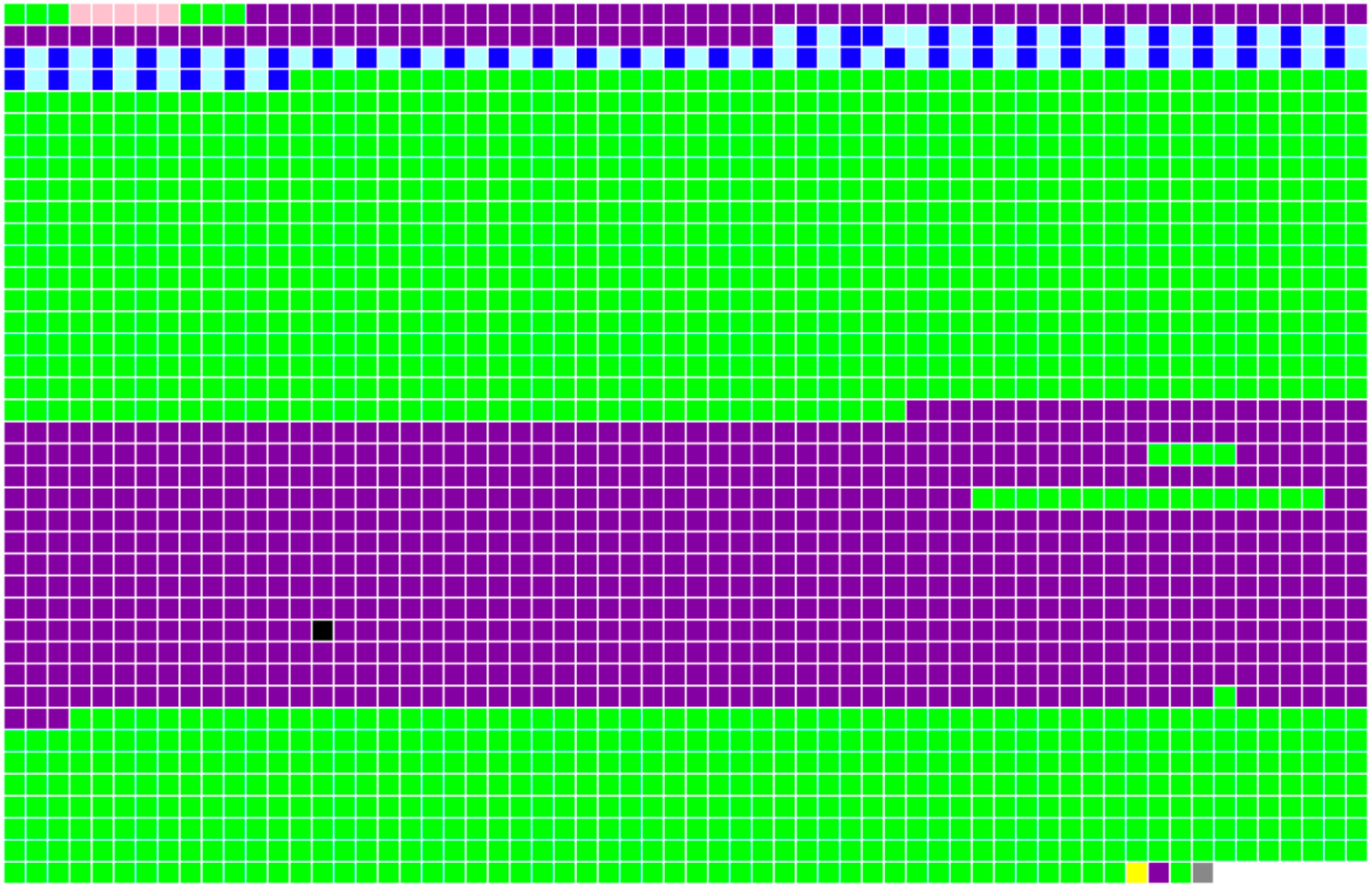
Inspectors

Compiled Method

address	4676295184
printString	Form >> #scaledByDisplayScaleFactor
header	10100000000000000000000000000011000000000000000110000011101
class	CompiledMethod
oopClassTag	3101
format	Compiled method (24)
hash	0
pinned	false
space	Old Space
immutable	false
selector	scaledByDisplayScaleFactor
methodClass	Form
numLiterals	7
numBytecodes	16
bytecode	an Array [16 items] (76 128 216 76 76 129 76 130 131 104 148 92 118 1 0 253)
literal 1	halt
literal 2	extent
literal 3	currentWorld
literal 4	displayScaleFactor
literal 5	scaledToSize:
literal 6	Instance of AdditionalMethodState
literal 7	Instance of GlobalVariable

Polyphemus Tools

Memory visualisation



1 pinned object

895 compiled method

51 class

5 special object

1 context

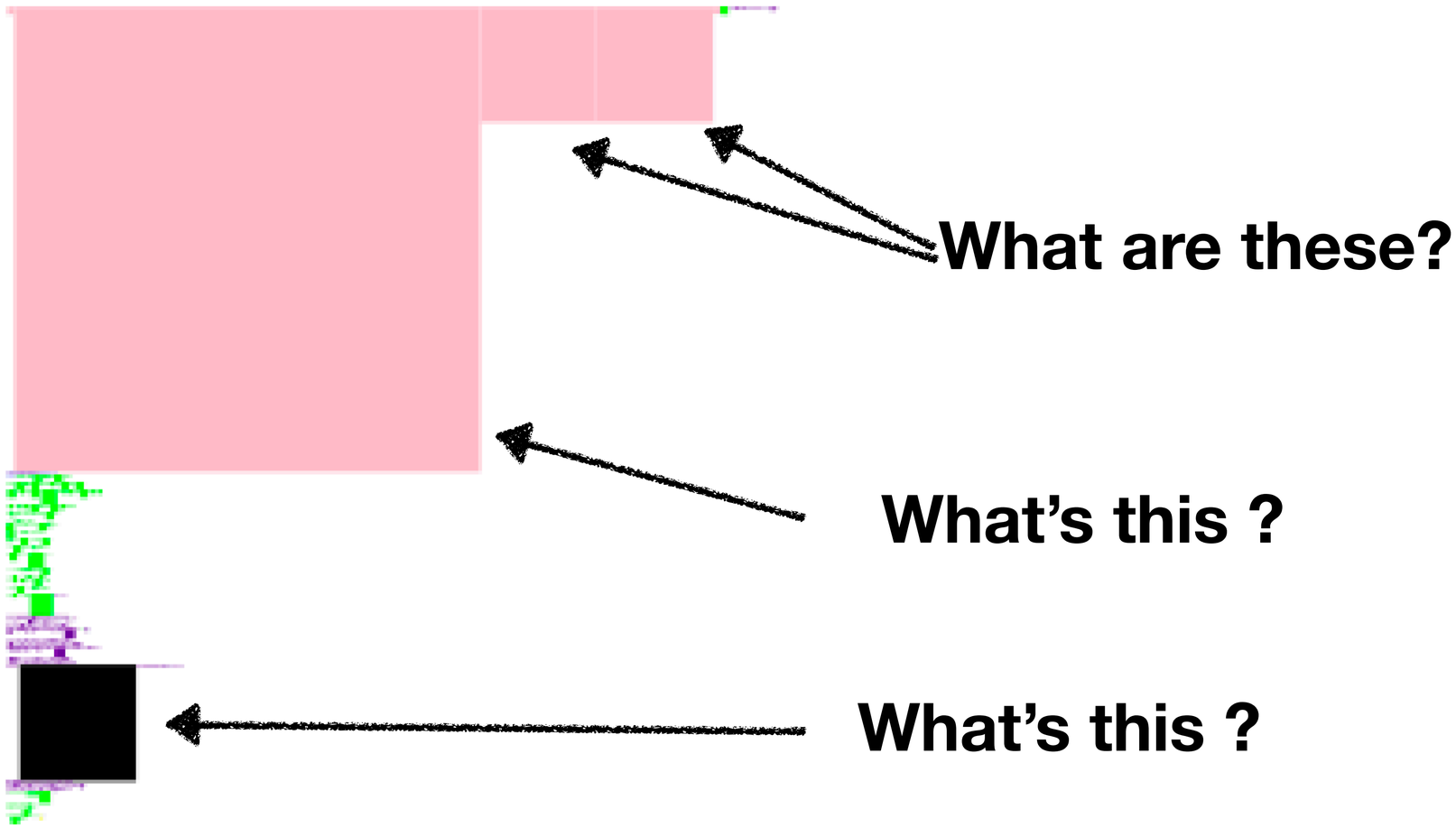
1 free chunk

1468 regular object

51 metaclass

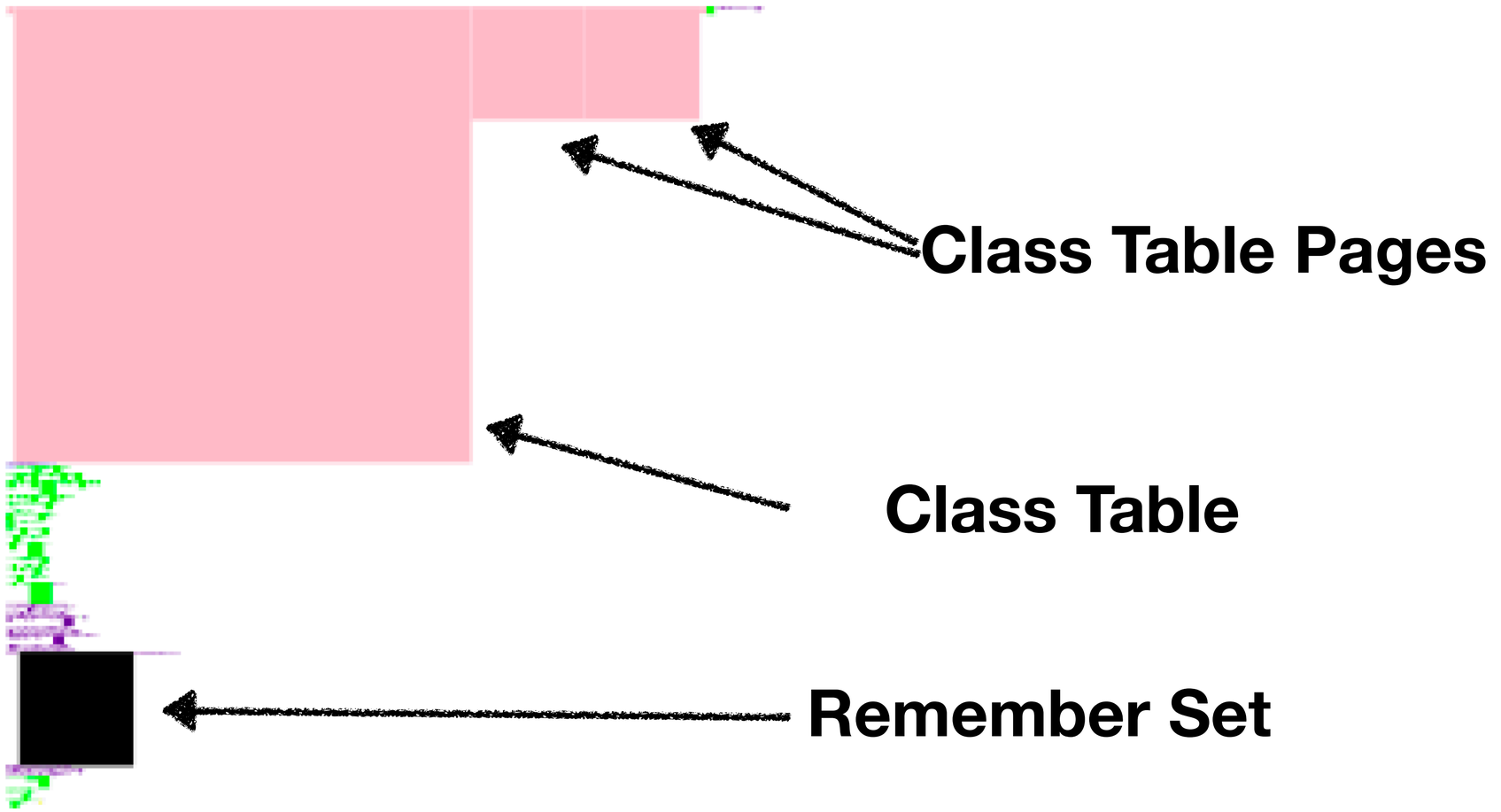
Polyphemus Tools

Evaluating Memory Size



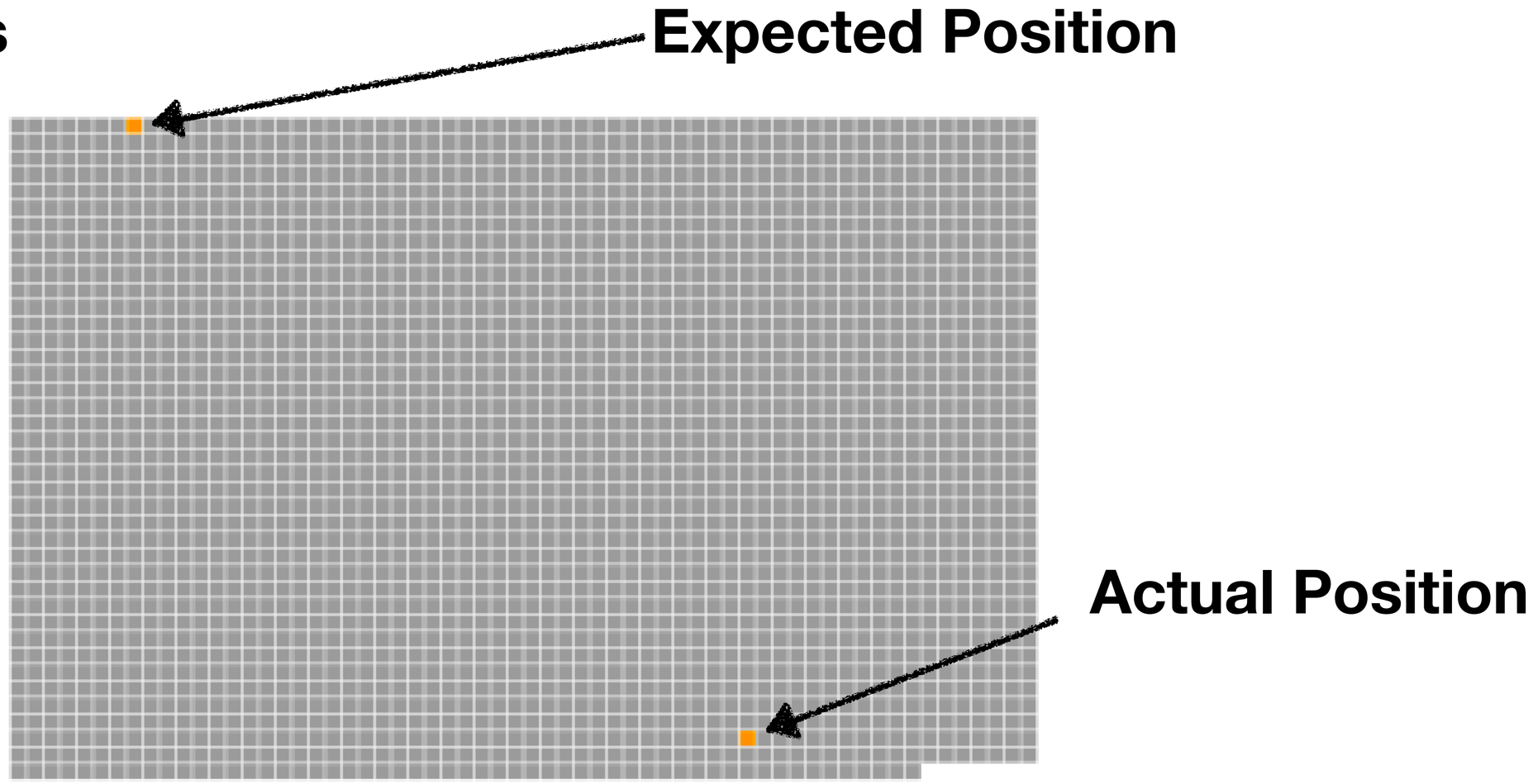
Polyphemus Tools

Evaluating Memory Size



Polyphemus Tools

Named Objects

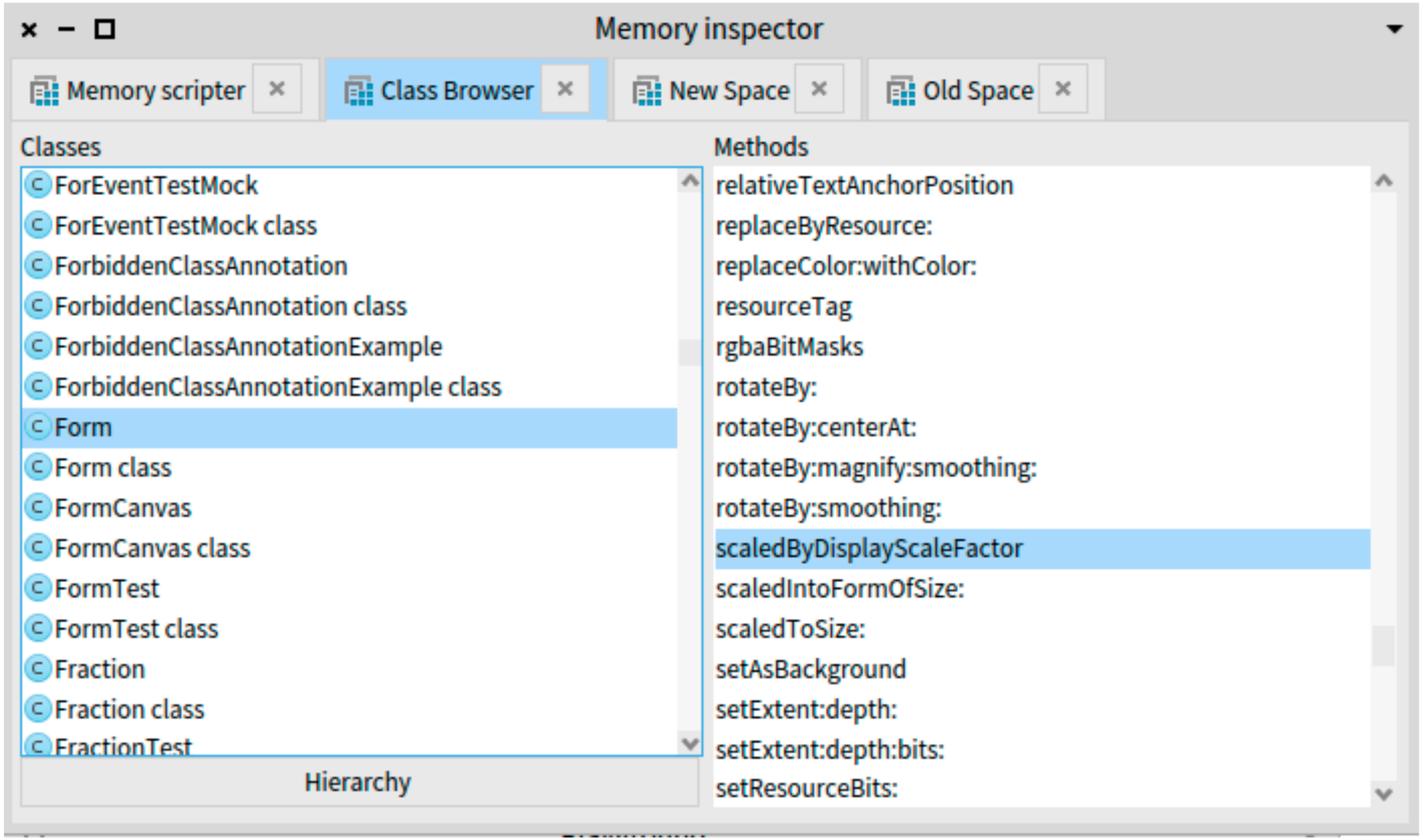


Selecting

```
[ :anOop | anOop numSlots = 60 ]
```

Polyphemus Tools

Memory Visualisation #2



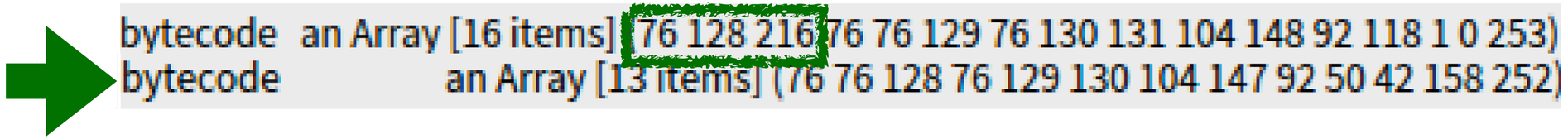
Real World Bug Fix #1

A Meta-Error

```
Form >> #scaledByDisplayScaleFactor
        self halt.
        ^ self scaledToSize: self extent * self currentWorld displayScaleFactor.
```

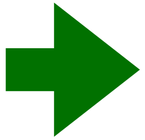
Real World Bug Fix #1

Investigating



bytecode	an Array [16 items] (76 128 216 76 76 129 76 130 131 104 148 92 118 1 0 253)
bytecode	an Array [13 items] (76 76 128 76 129 130 104 147 92 50 42 158 252)

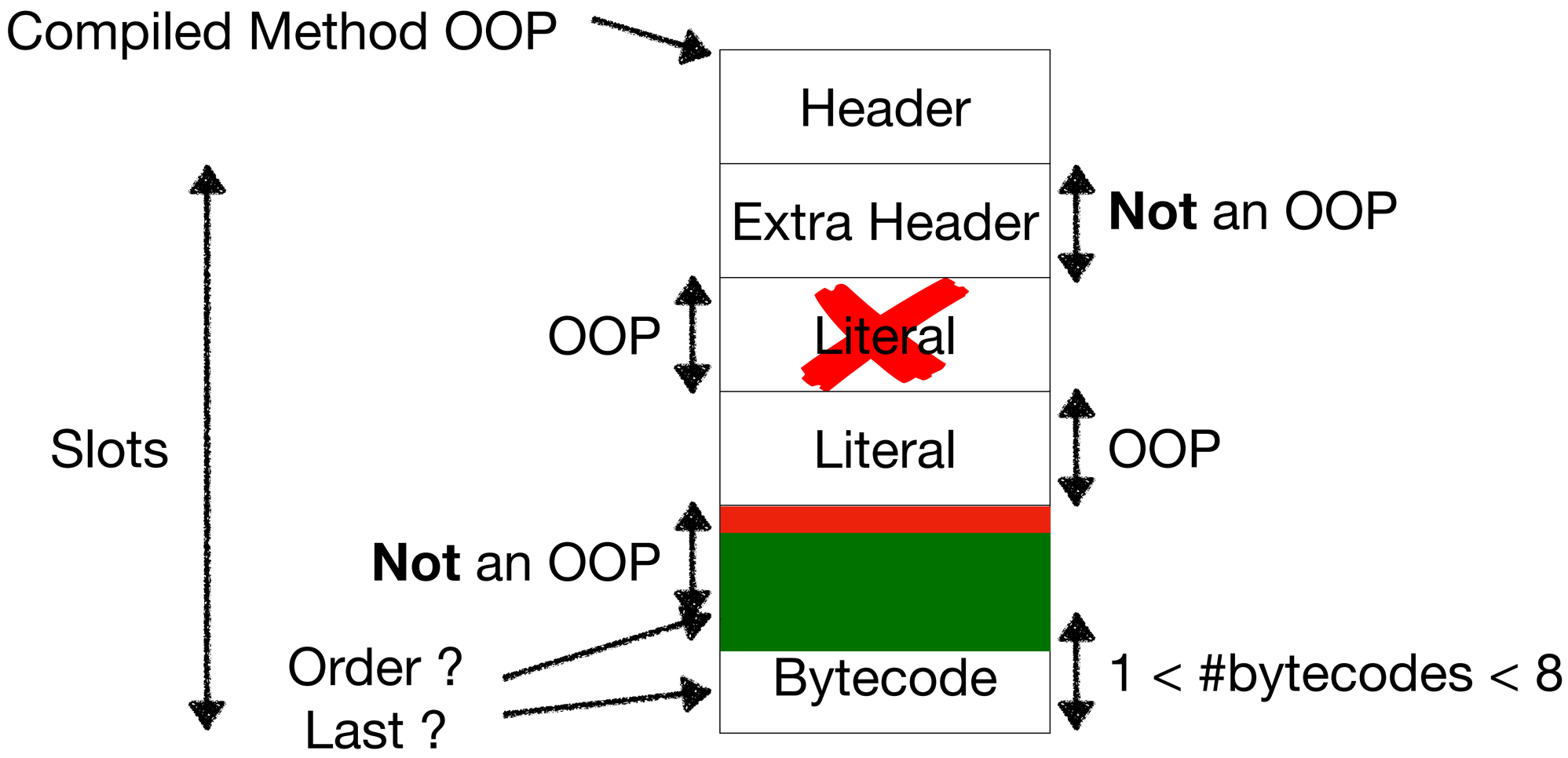
literal 1	halt
literal 2	extent
literal 3	currentWorld
literal 4	displayScaleFactor
literal 5	scaledToSize:
literal 6	scaledByDisplayScaleFactor
literal 7	Instance of GlobalVariable



literal 1	extent
literal 2	currentWorld
literal 3	displayScaleFactor
literal 4	scaledToSize:
literal 5	scaledByDisplayScaleFactor
literal 6	Instance of GlobalVariable

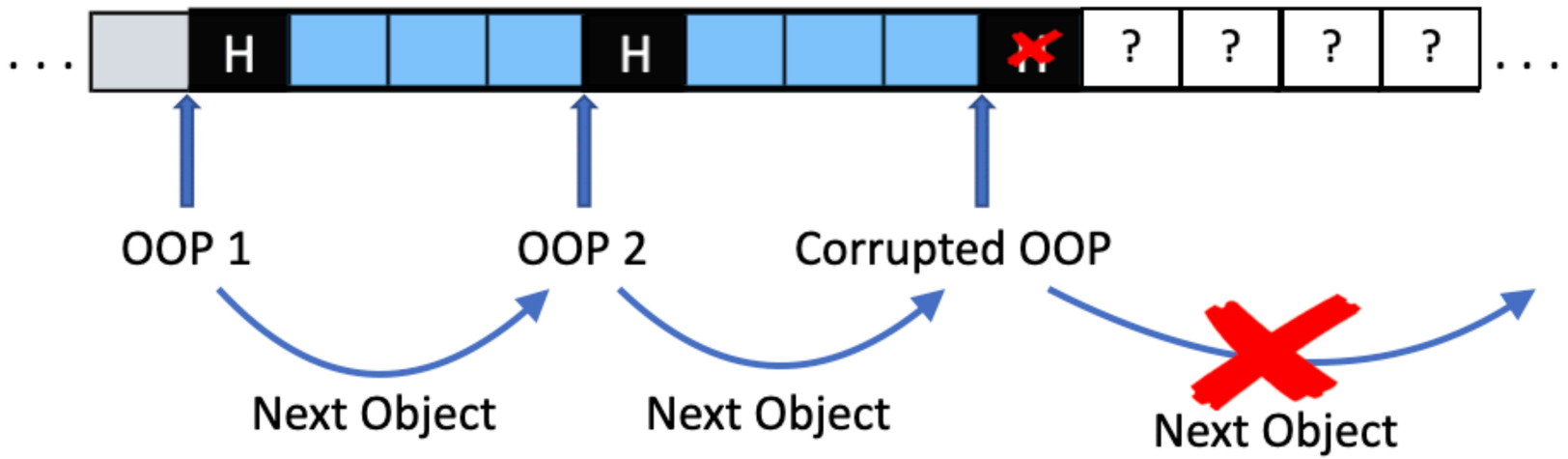
Real World Bug Fix #1

Method Patching Analysis



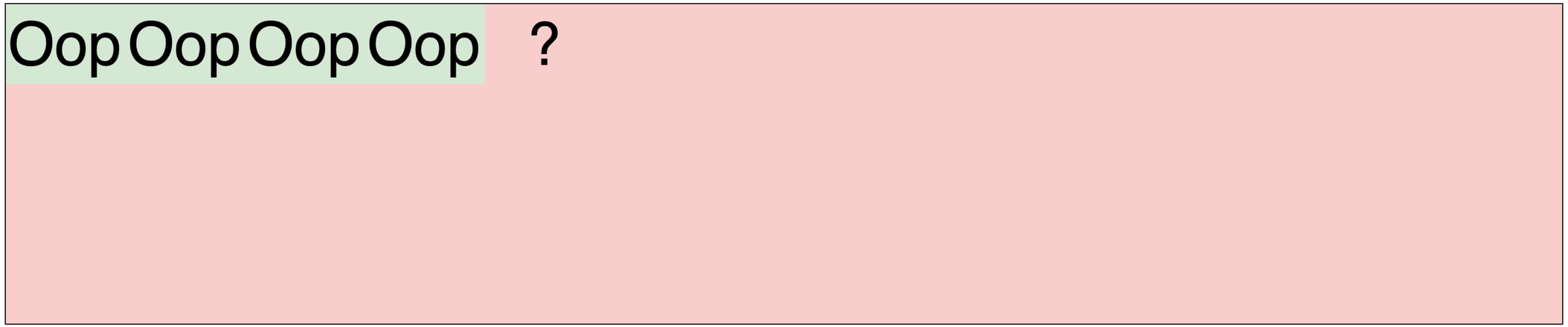
Real World Bug Fix #2

Iterating the Corrupted Memory



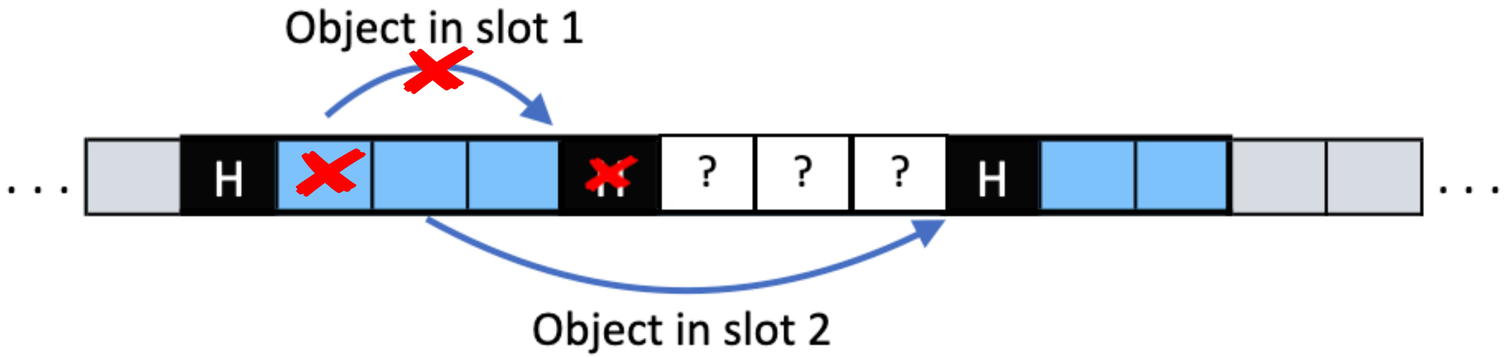
Real World Bug Fix #2

We don’t know what’s after



Real World Bug Fix #2

Recovering Objects



Real World Bug Fix #2

Tracking The Corruption

Oop	Oop	Oop	Oop	Oop	C	Oop	Oop	Oop	Oop
Oop	Oop	Oop	F	Oop	Oop	Oop	Oop	Oop	Oop
Oop	Oop	C	Oop	Oop	Oop	F	Oop	Oop	Oop
Oop	F	Oop	F	Oop	C	Oop	Oop	Oop	Oop
Oop	Oop	F	Oop	Oop	Oop	Oop	F	Oop	Oop

Real World Bug Fix #2

Cleansing The Corruption

Oop	Oop	Oop	Oop	Oop	F	Oop	Oop	Oop	Oop
Oop	Oop	Oop	F	Oop	Oop	Oop	Oop	Oop	Oop
Oop	Oop	F	Oop	Oop	Oop	F	Oop	Oop	Oop
Oop	F	Oop	F	Oop	F	Oop	Oop	Oop	Oop
Oop	Oop	F	Oop	Oop	Oop	Oop	F	Oop	Oop

Real World Bug Fix #2

Corruption Cleansing Analysis

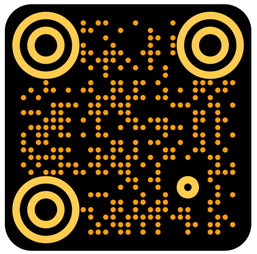
- Objects’ slots iteration
- Reference patching
- Focus on learning rather than how to do the previous items

Conclusion

Github

- Tooling at the VM level is difficult
- LLOOPs bridge knowledge gaps
  - Language Level entities
  - Identifying and typing OOPs
  - Mix Language and VM level informations

Paper



Pierre Misse-Chanabier
pierre_misse25@msn.com
github.com/hogoww
Discord tag: hogo#8547