Memory Profiling in Pharo

Sebastian JORDAN-MONTAÑO

sebastian.jordan@inria.fr

Inria, Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL

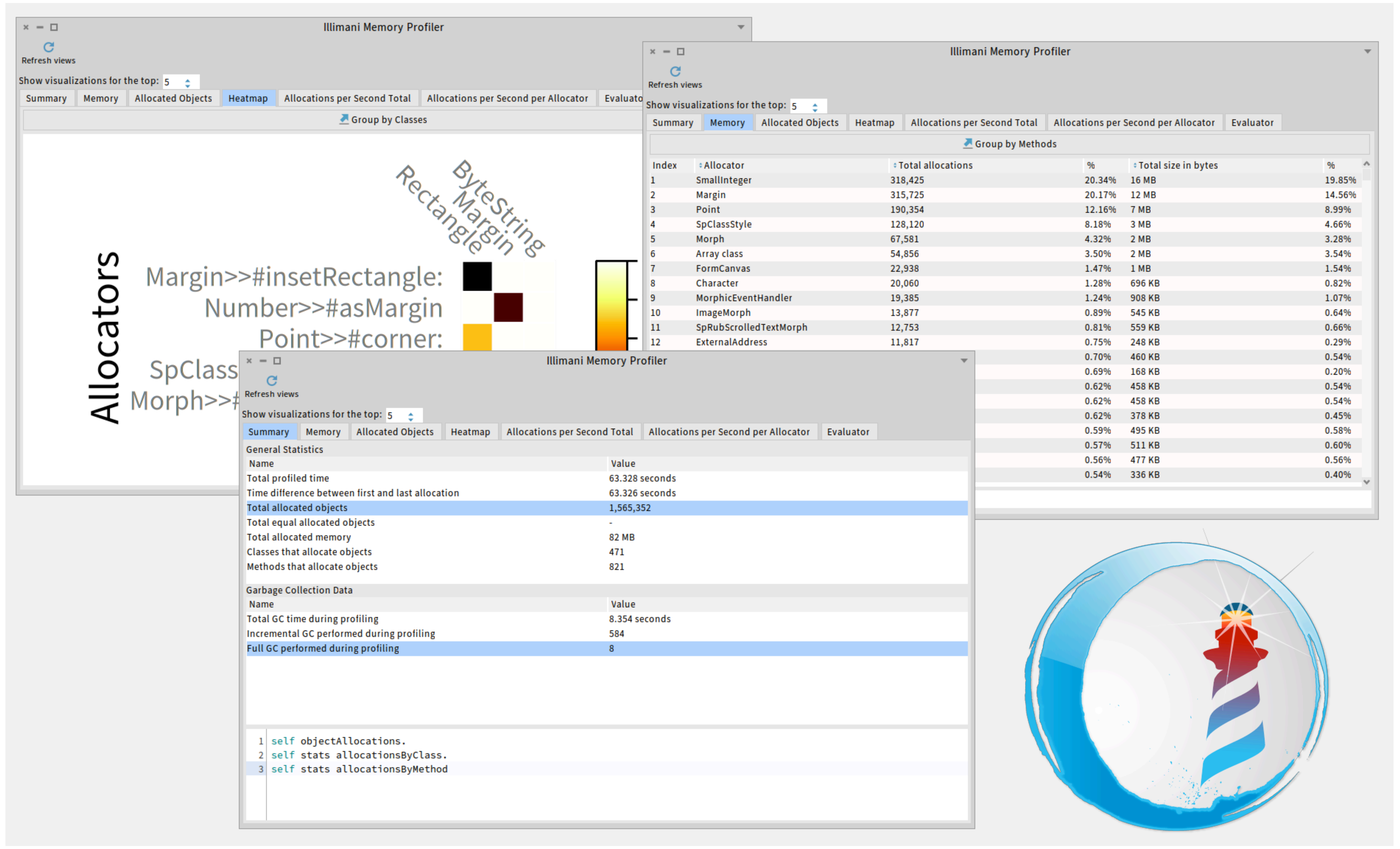
August 2023

About me

- **Bachelor's:** Software Engineer, UCB, Bolivia
- **Master's:** Software Engineering, UL, France
- **PhD:** Starting in profiling
- **Interests:** Music (progressive rock, Charly), languages, sports, Pharo

Illimani: a Pharo memory profiler

- Open-source MIT license
- Detects object allocation sites
- Tracks object lifetimes
- Allocation matrix
- Unmodified VM
- Density chart
- Memory consumption tables
- Rich object-oriented model



Index	Allocator	Total allocations	%	Total size in bytes	%
1	SmallInteger	318,425	20.34%	16 MB	19.85%
2	Margin	315,725	20.17%	12 MB	14.56%
3	Point	190,354	12.16%	7 MB	8.99%
4	SpClassStyle	128,120	8.18%	3 MB	4.66%
5	Morph	67,581	4.32%	2 MB	3.28%
6	Array class	54,856	3.50%	2 MB	3.54%
7	FormCanvas	22,938	1.47%	1 MB	1.54%
8	Character	20,060	1.28%	696 KB	0.82%
9	MorphicEventHandler	19,385	1.24%	908 KB	1.07%
10	ImageMorph	13,877	0.89%	545 KB	0.64%
11	SpRubScrolledTextMorph	12,753	0.81%	559 KB	0.66%
12	ExternalAddress	11,817	0.75%	248 KB	0.29%
			0.70%	460 KB	0.54%
			0.69%	168 KB	0.20%
			0.62%	458 KB	0.54%
			0.62%	458 KB	0.54%
			0.62%	378 KB	0.45%
			0.59%	495 KB	0.58%
			0.57%	511 KB	0.60%
			0.56%	477 KB	0.56%
			0.54%	336 KB	0.40%

General Statistics

Name	Value
Total profiled time	63.328 seconds
Time difference between first and last allocation	63.326 seconds
Total allocated objects	1,565,352
Total equal allocated objects	-
Total allocated memory	82 MB
Classes that allocate objects	471
Methods that allocate objects	821

Garbage Collection Data

Name	Value
Total GC time during profiling	8.354 seconds
Incremental GC performed during profiling	584
Full GC performed during profiling	8

```
self objectAllocations.
self stats allocationsByClass.
self stats allocationsByMethod
```

github.com/jordanmontt/illimani-memory-profiler

Object allocation sites

We define an object allocation site as the textual location in the source code where the object was created [1]

```
AthensTextScanner >> initialize

    lines := OrderedCollection new
    ...
```

Allocation site

[1] **Memento Mori: Dynamic Allocation-Site-Based Optimizations**

Daniel Clifford   Hannes Payer   Michael Stanton   Ben L. Titzer

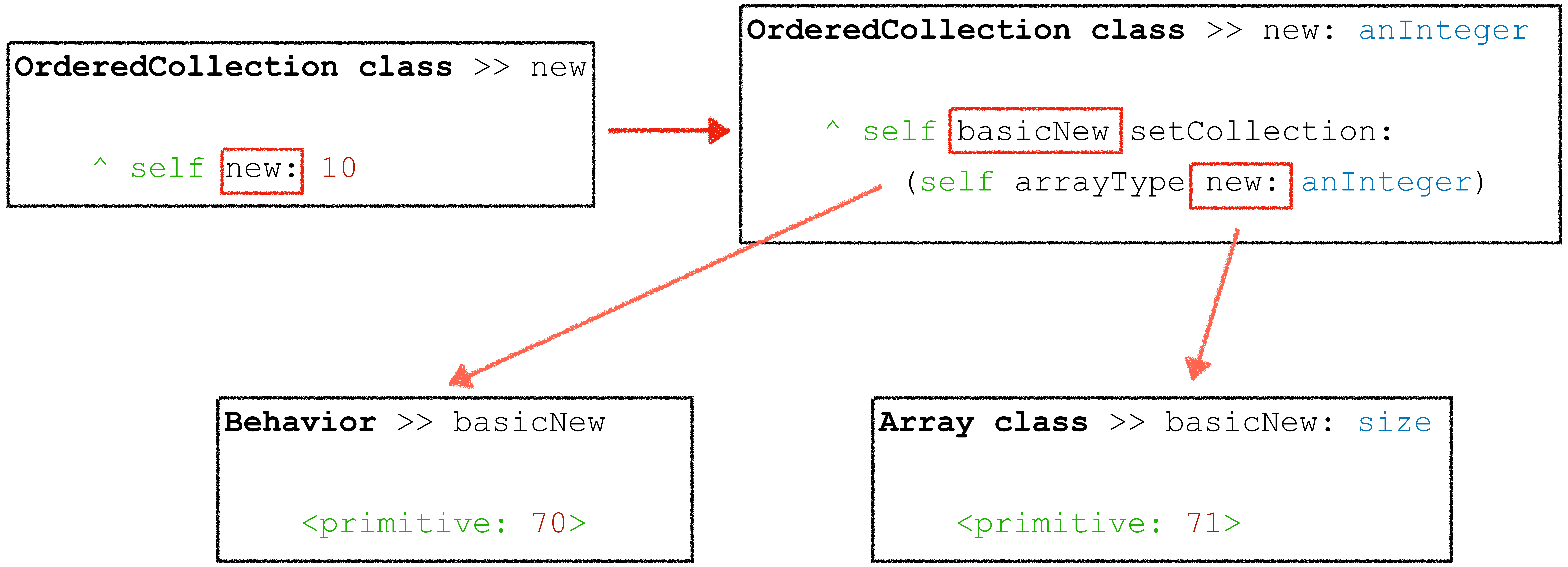
Capture object allocation sites

In Pharo, almost all computations are done by sending a message. This is also true when allocating objects.

Allocating an object

```
OrderedCollection new
```

Allocating an object



```
OrderedCollection class >> new

    ^ self new: 10
```

```
OrderedCollection class >> new: anInteger

    ^ self basicNew setCollection:
      (self arrayType new: anInteger)
```

```
Behavior >> basicNew

    <primitive: 70>
```

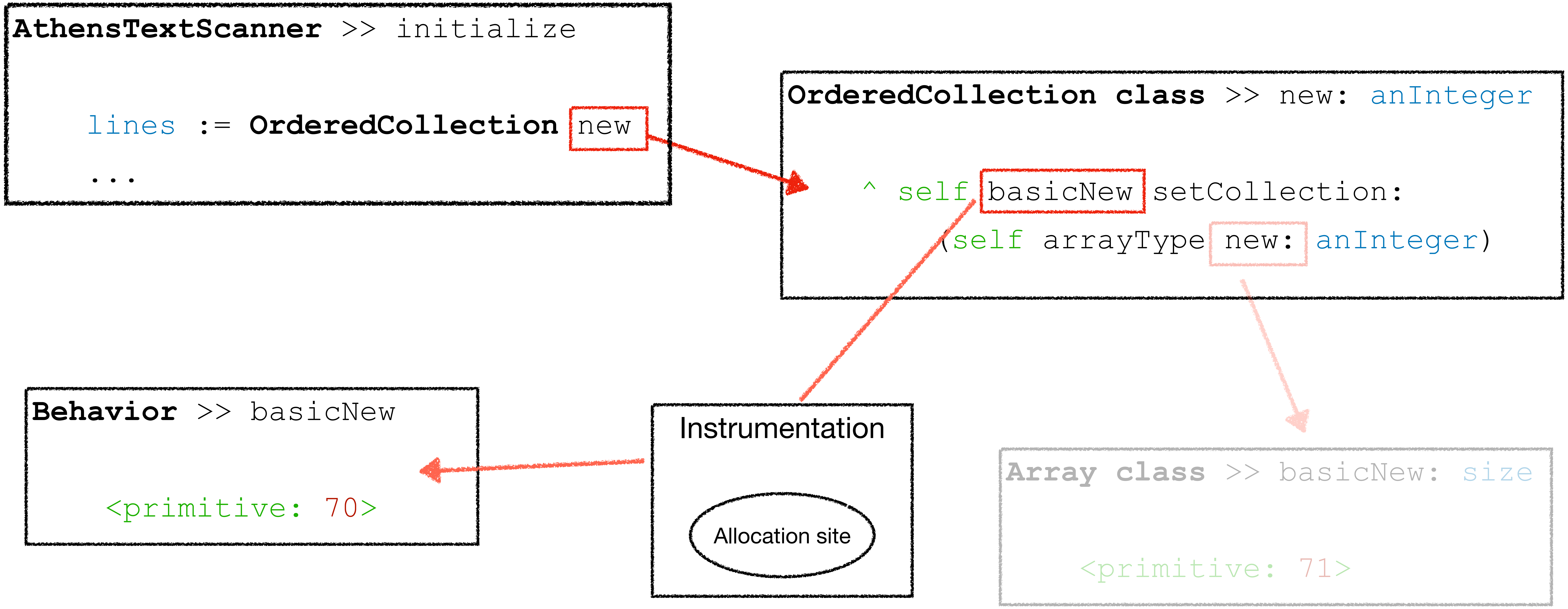
```
Array class >> basicNew: size

    <primitive: 71>
```

Instrumenting object allocation sites in Pharo

We've instrumented 3 methods that allocate objects to capture when they will be called and then filter the execution stack.

- `Behavior >> basicNew`
- `Behavior >> basicNew:`
- `Array class >> new:`

Capture object allocation sites



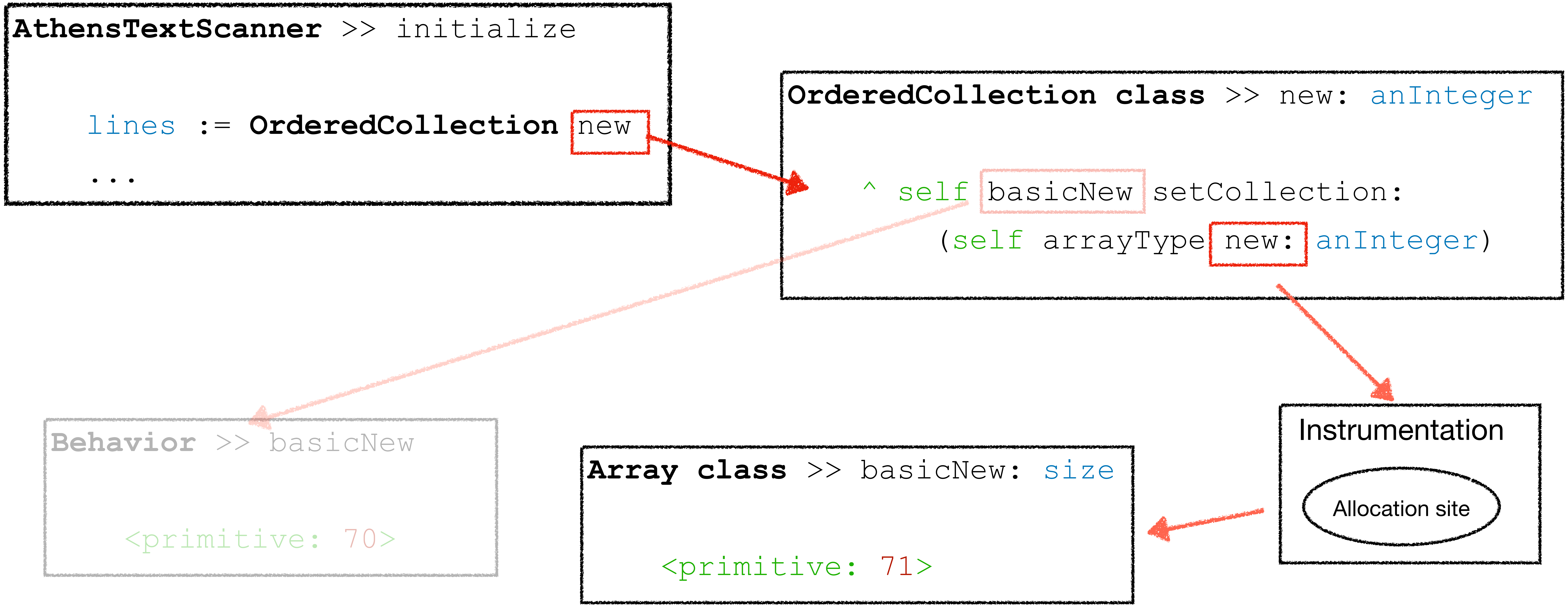
```
AthensTextScanner >> initialize

    lines := OrderedCollection new
    ...
```

```
OrderedCollection class >> new: anInteger

    ^ self basicNew setCollection:
        (self arrayType new: anInteger)
```

```
Behavior >> basicNew

    <primitive: 70>
```

```
Array class >> basicNew: size

    <primitive: 71>
```

Allocating an object



```
AthensTextScanner >> initialize

    lines := OrderedCollection new
    ...
```

```
OrderedCollection class >> new: anInteger

    ^ self basicNew setCollection:
        (self arrayType new: anInteger)
```

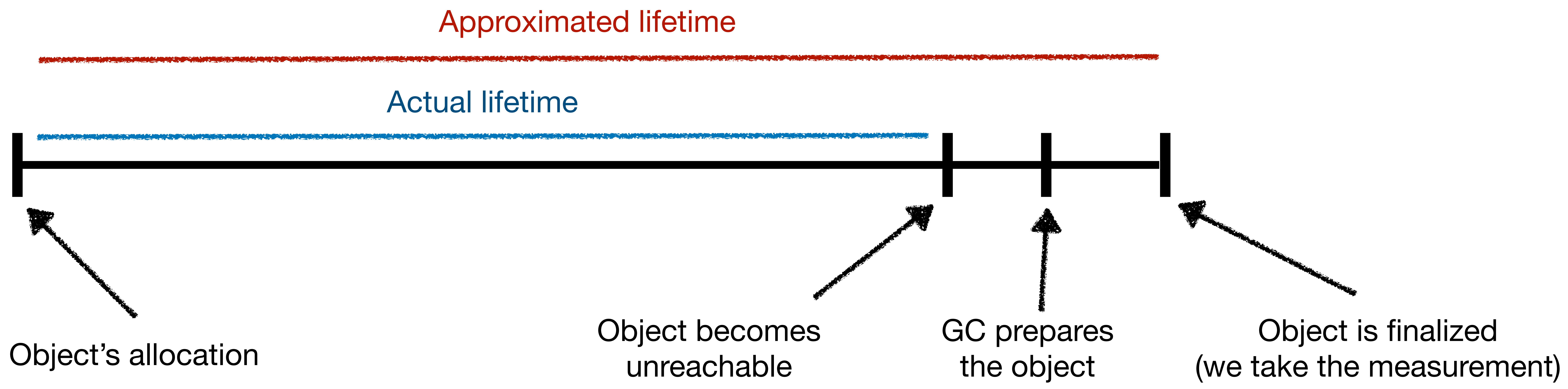
```
Behavior >> basicNew

    <primitive: 70>
```

```
Array class >> basicNew: size

    <primitive: 71>
```

An object’s (approximated) lifetime

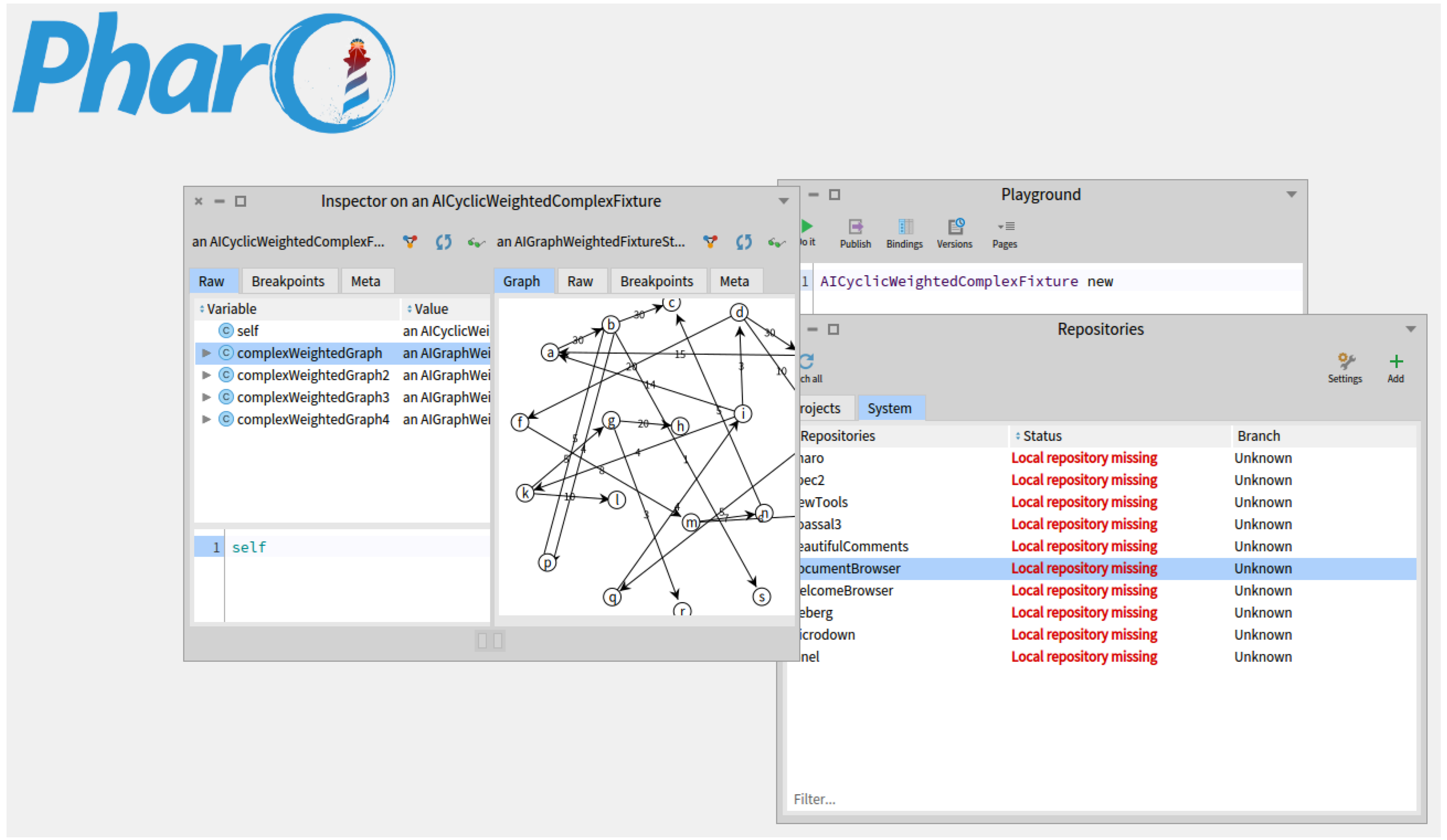
$$\boxed{\textit{object'sLifetime} = \textit{finalizationTime} - \textit{allocationTime}}$$

An object’s lifetime



Case study 1: Morphic

- Profiled the allocations of type `Color`
- We opened 30 Pharo tools (Inspector, Playground and Iceberg) and we let them render for 100 rendering cycles



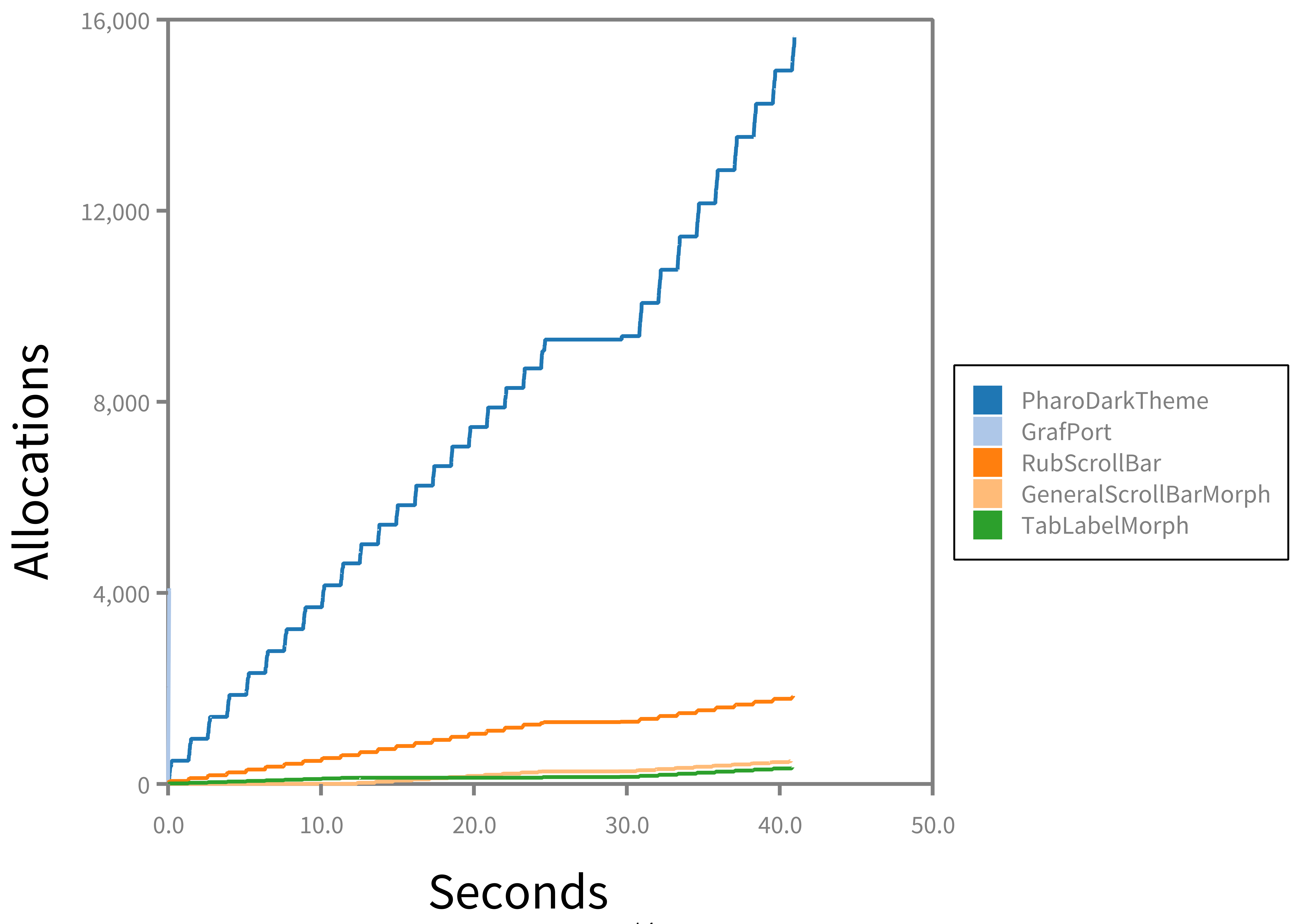
Variable	Value
self	an AICyclicWei
complexWeightedGraph	an AIGraphWei
complexWeightedGraph2	an AIGraphWei
complexWeightedGraph3	an AIGraphWei
complexWeightedGraph4	an AIGraphWei

```
self
```

```
AICyclicWeightedComplexFixture new
```

Repositories	Status	Branch
haro	Local repository missing	Unknown
pec2	Local repository missing	Unknown
ewTools	Local repository missing	Unknown
oassal3	Local repository missing	Unknown
eautifulComments	Local repository missing	Unknown
ocumentBrowser	Local repository missing	Unknown
elcomeBrowser	Local repository missing	Unknown
eberg	Local repository missing	Unknown
icrodown	Local repository missing	Unknown
nel	Local repository missing	Unknown

Allocations over time



Morphic Analysis

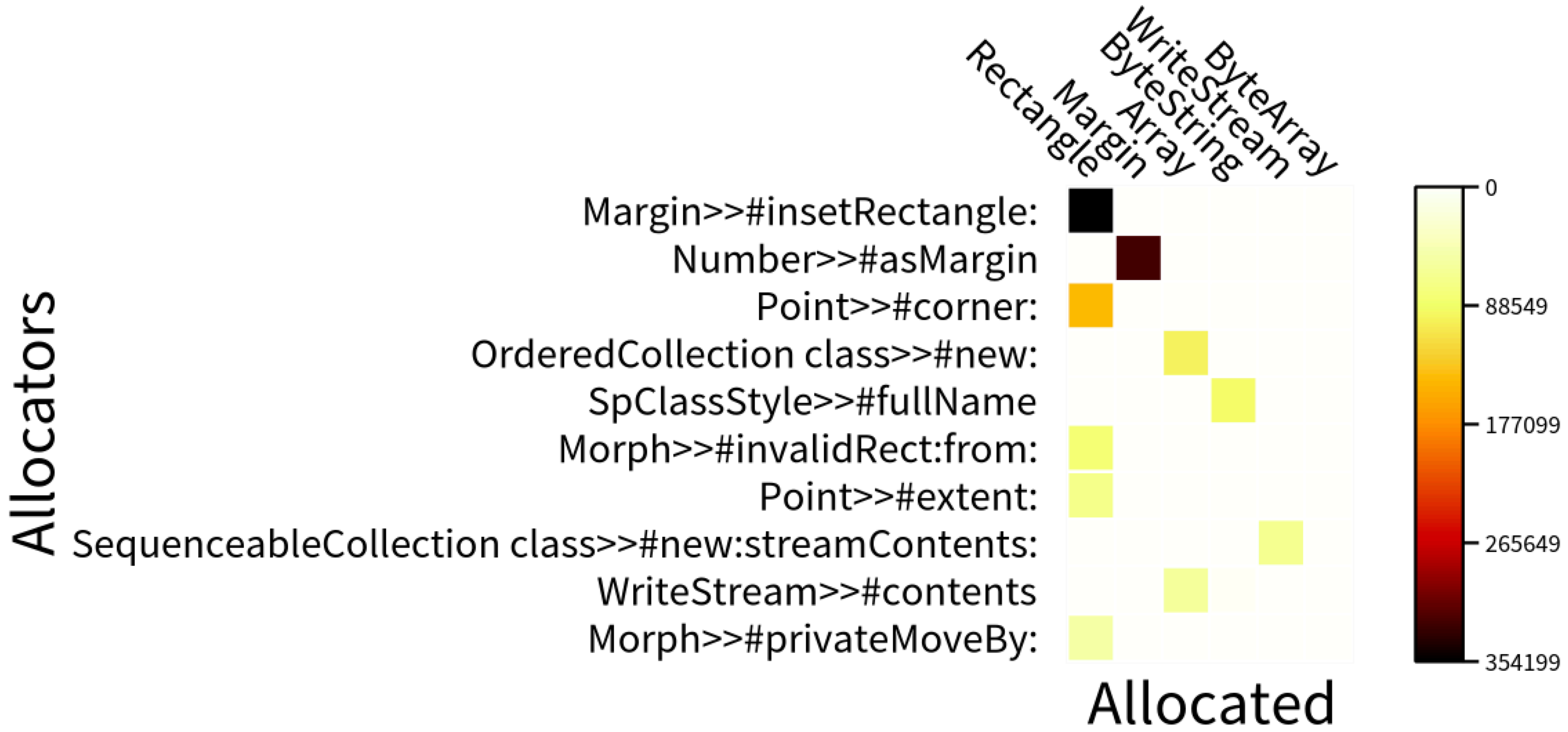
Allocator class	Allocated colors	%
PharoDarkTheme	15,629	66%
GrafPort	4,096	17%
RubScrollBar	1,842	8%
GeneralScrollBarMorph	480	2%
TabLabelMorph	346	1%
Rest of the classes	1293	2%

We have identified an object allocation site in the class `PharoDarkTheme` that allocated 66% of all the allocated colors with 99,9% redundant allocations.

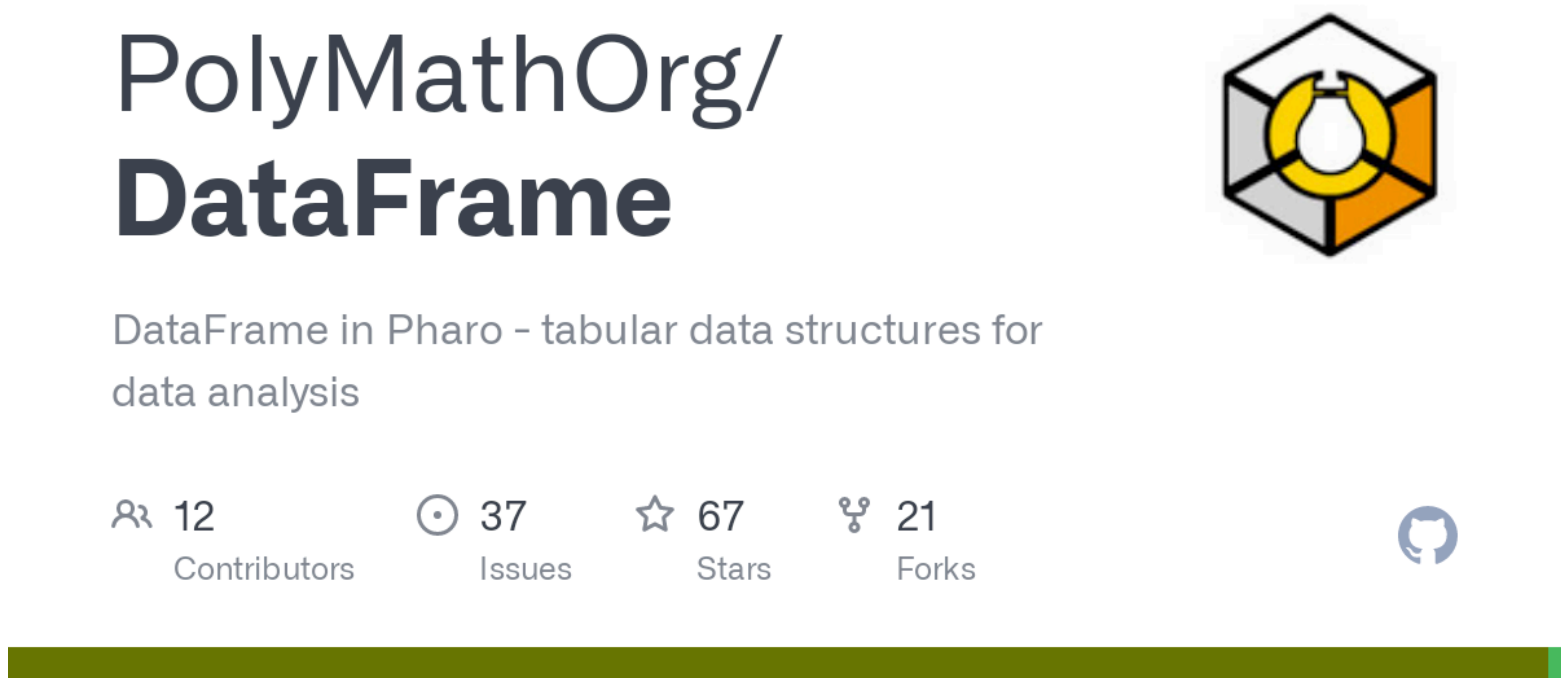
Morphic after the fix

Before
After
Allocations
16,000
12,000
8,000
4,000
0
5,000
4,000
3,000
2,000
1,000
0
0.0
10.0
20.0
30.0
40.0
50.0
Seconds
GrafPort
RubScrollBar
GeneralScrollBarMorph
TabLabelMorph
AlignmentMorph

Detecting other allocation sites



Case study 2: DataFrame



PolyMathOrg/
DataFrame

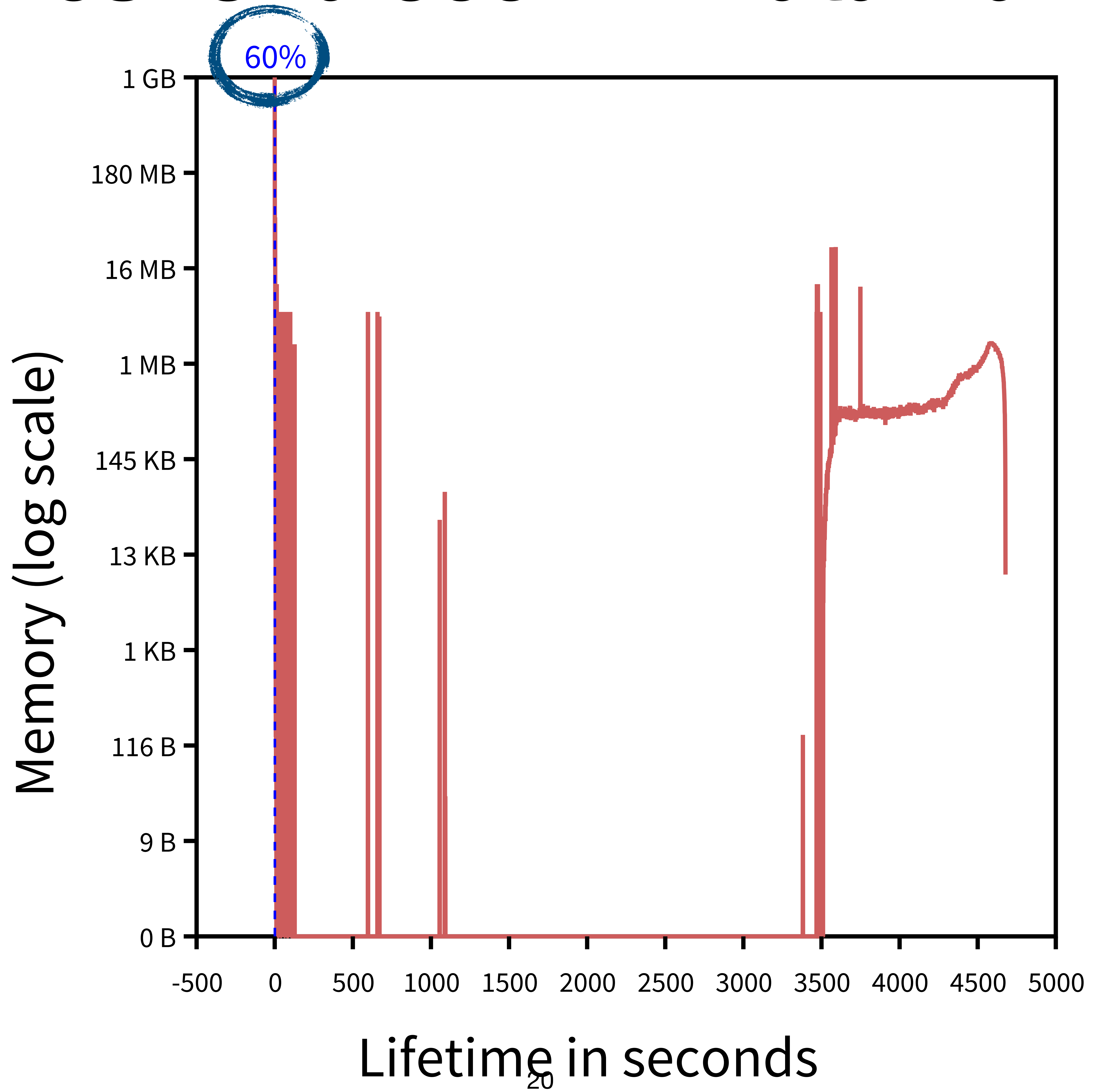
DataFrame in Pharo - tabular data structures for data analysis

12	37	67	21
Contributors	Issues	Stars	Forks

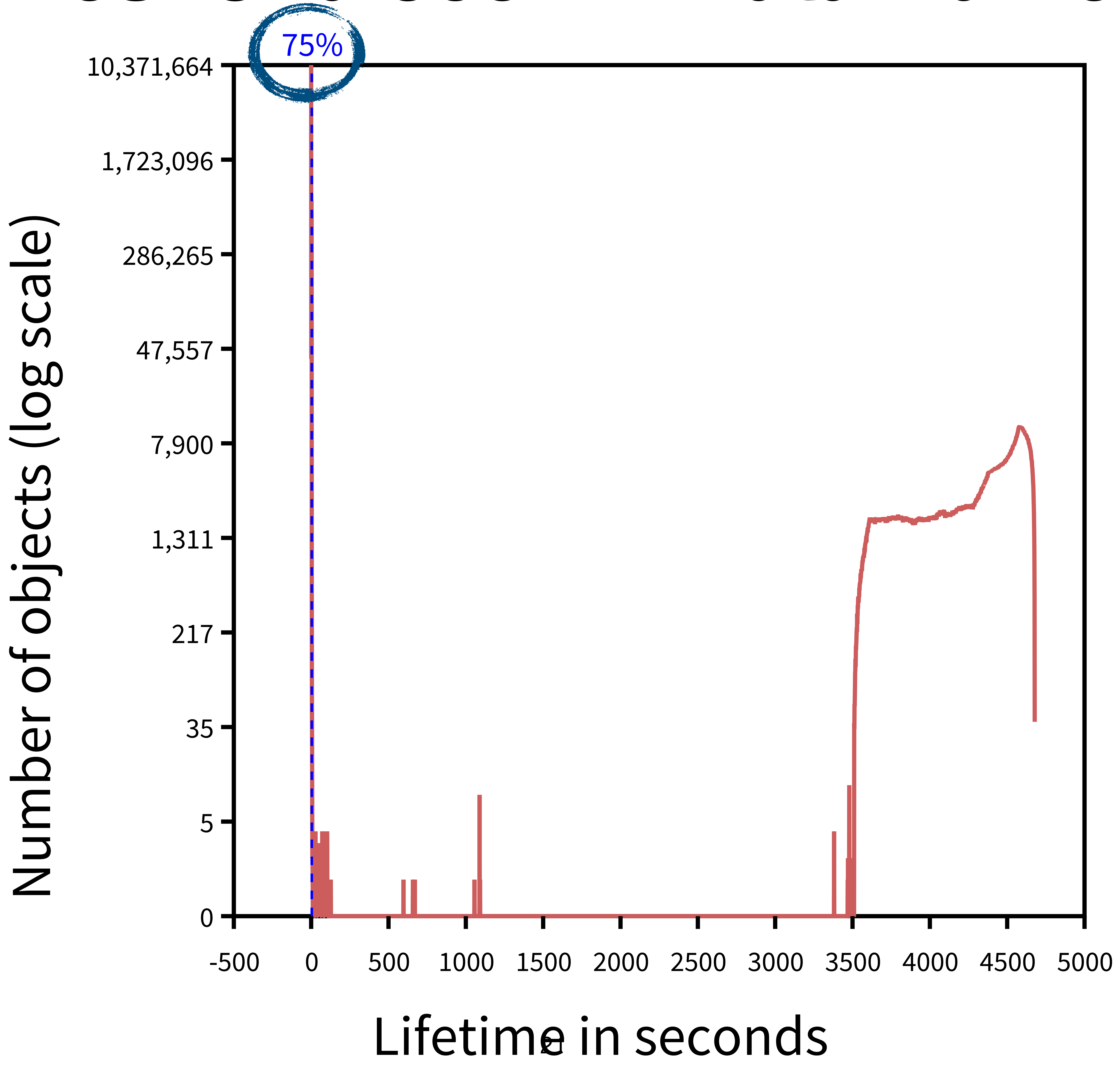
Benchmarking DataFrame

Dataset	# of scavengers	# of full GCs	GC time	Total time	GC time in %
500 MB	266	18	11 sec	71 sec	15%
1.6 GB	304	36	60 sec	248 sec	22%
3.1 GB	1143	309	3793 sec	4265 sec	89%

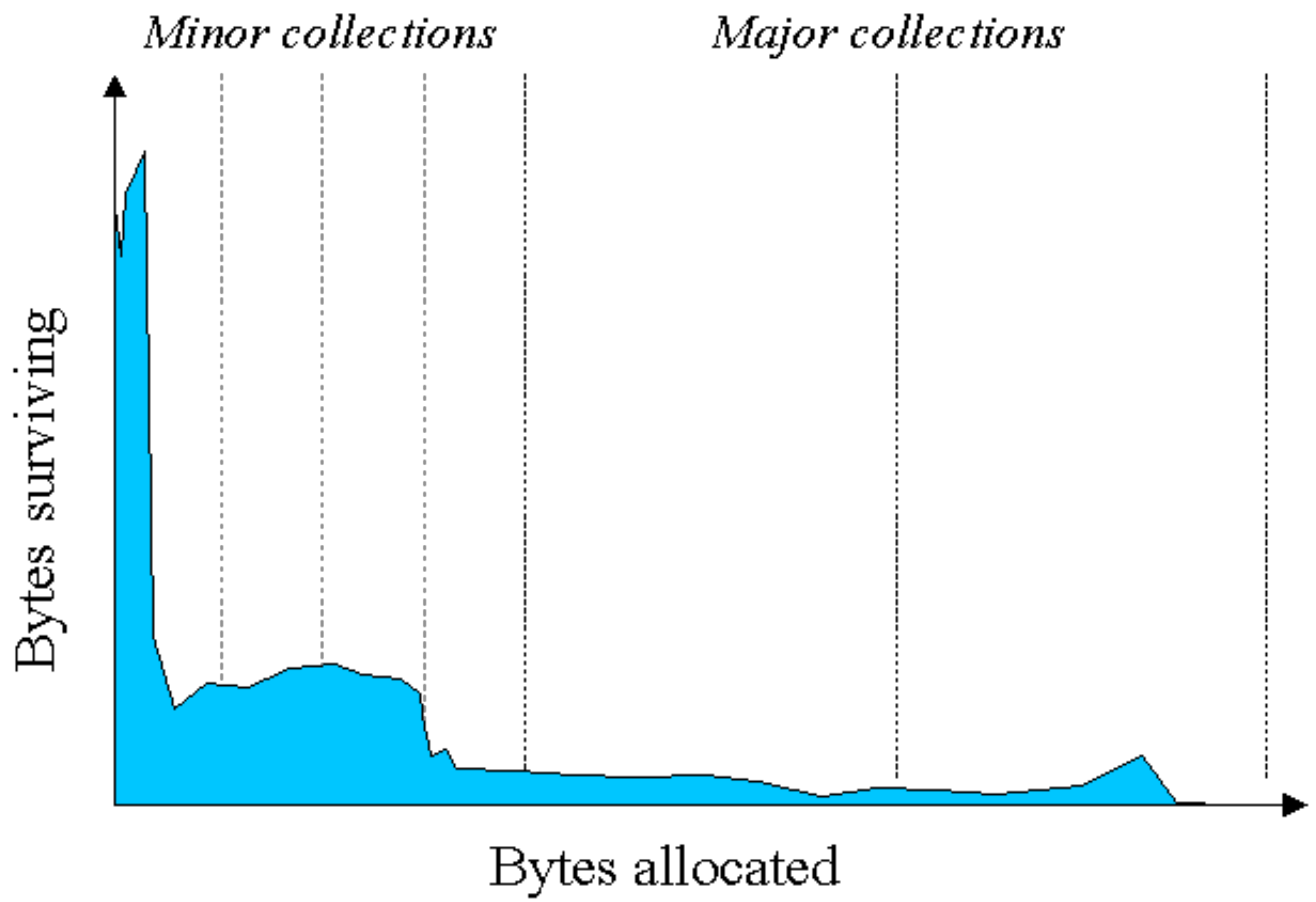
Object lifetimes for a 500MB DataFrame (memory)



Object lifetimes for a 500MB DataFrame (# objects)



A common object lifetime distribution



Source: oracle.com

DataFrame performance improvements

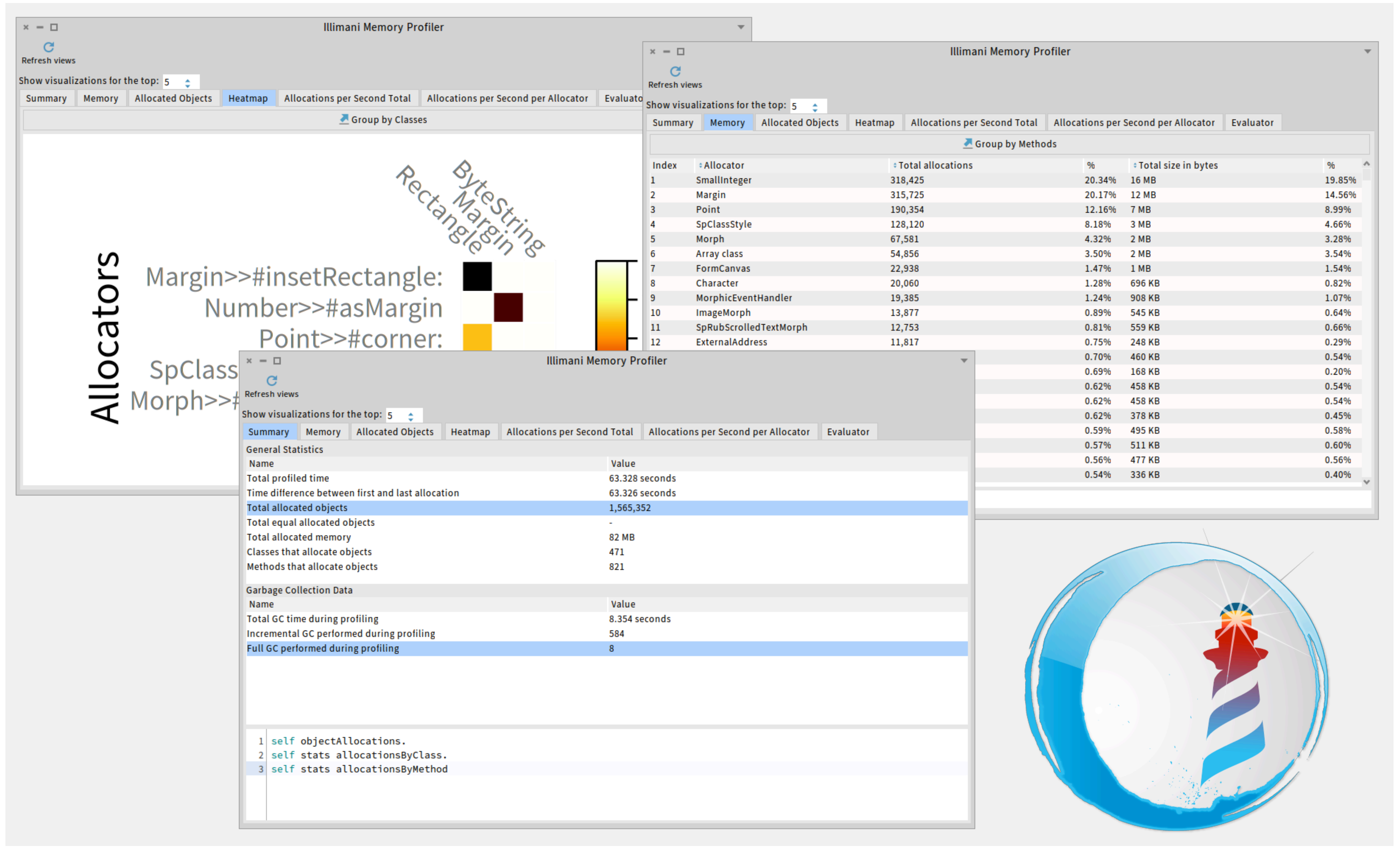
GC Configuration	GC spent time	Total execution time	Improved performance
Default	58 min 18 sec	1 hour 6 min 18 sec	1×
Configuration 1	9 min 41 sec	17 min 46 sec	3.7×
Configuration 2	4 min 57 sec	12 min 54 sec	5.1×
Configuration 3	5 min 8 sec	13 min 2 sec	5.1×
Configuration 4	2 min 42 sec	10 min 37 sec	6.2×
Configuration 5	1 min 47 sec	9 min 42 sec	6.8×

The future

- Study the precision of the approximated lifetimes.
- Calculating the object lifetimes at virtual machine level.
- Dynamic optimizations based on allocation sites.

Illimani: a Pharo memory profiler

- Open-source MIT license
- Detects object allocation sites
- Tracks object lifetimes
- Allocation matrix
- Unmodified VM
- Density chart
- Memory consumption tables
- Rich object-oriented model



Index	Allocator	Total allocations	%	Total size in bytes	%
1	SmallInteger	318,425	20.34%	16 MB	19.85%
2	Margin	315,725	20.17%	12 MB	14.56%
3	Point	190,354	12.16%	7 MB	8.99%
4	SpClassStyle	128,120	8.18%	3 MB	4.66%
5	Morph	67,581	4.32%	2 MB	3.28%
6	Array class	54,856	3.50%	2 MB	3.54%
7	FormCanvas	22,938	1.47%	1 MB	1.54%
8	Character	20,060	1.28%	696 KB	0.82%
9	MorphicEventHandler	19,385	1.24%	908 KB	1.07%
10	ImageMorph	13,877	0.89%	545 KB	0.64%
11	SpRubScrolledTextMorph	12,753	0.81%	559 KB	0.66%
12	ExternalAddress	11,817	0.75%	248 KB	0.29%
			0.70%	460 KB	0.54%
			0.69%	168 KB	0.20%
			0.62%	458 KB	0.54%
			0.62%	458 KB	0.54%
			0.62%	378 KB	0.45%
			0.59%	495 KB	0.58%
			0.57%	511 KB	0.60%
			0.56%	477 KB	0.56%
			0.54%	336 KB	0.40%

General Statistics

Name	Value
Total profiled time	63.328 seconds
Time difference between first and last allocation	63.326 seconds
Total allocated objects	1,565,352
Total equal allocated objects	-
Total allocated memory	82 MB
Classes that allocate objects	471
Methods that allocate objects	821

Garbage Collection Data

Name	Value
Total GC time during profiling	8.354 seconds
Incremental GC performed during profiling	584
Full GC performed during profiling	8

```
self objectAllocations.
self stats allocationsByClass.
self stats allocationsByMethod
```

Sebastian JORDAN MONTAÑO

sebastian.jordan@inria.fr

github.com/jordanmontt/illimani-memory-profiler