

# Profilage en Pharo

Étudiant : Sebastian JORDAN MONTAÑO

Superviseurs entreprise :

- Stéphane Ducasse
- Guillermo Polito



Juillet 2023

# L'équipe de recherche EVREF

- Implémentation de langages de programmation
- Méta programmation
- Transformation du code
- Migration des logiciels
- Pharo



# Plan

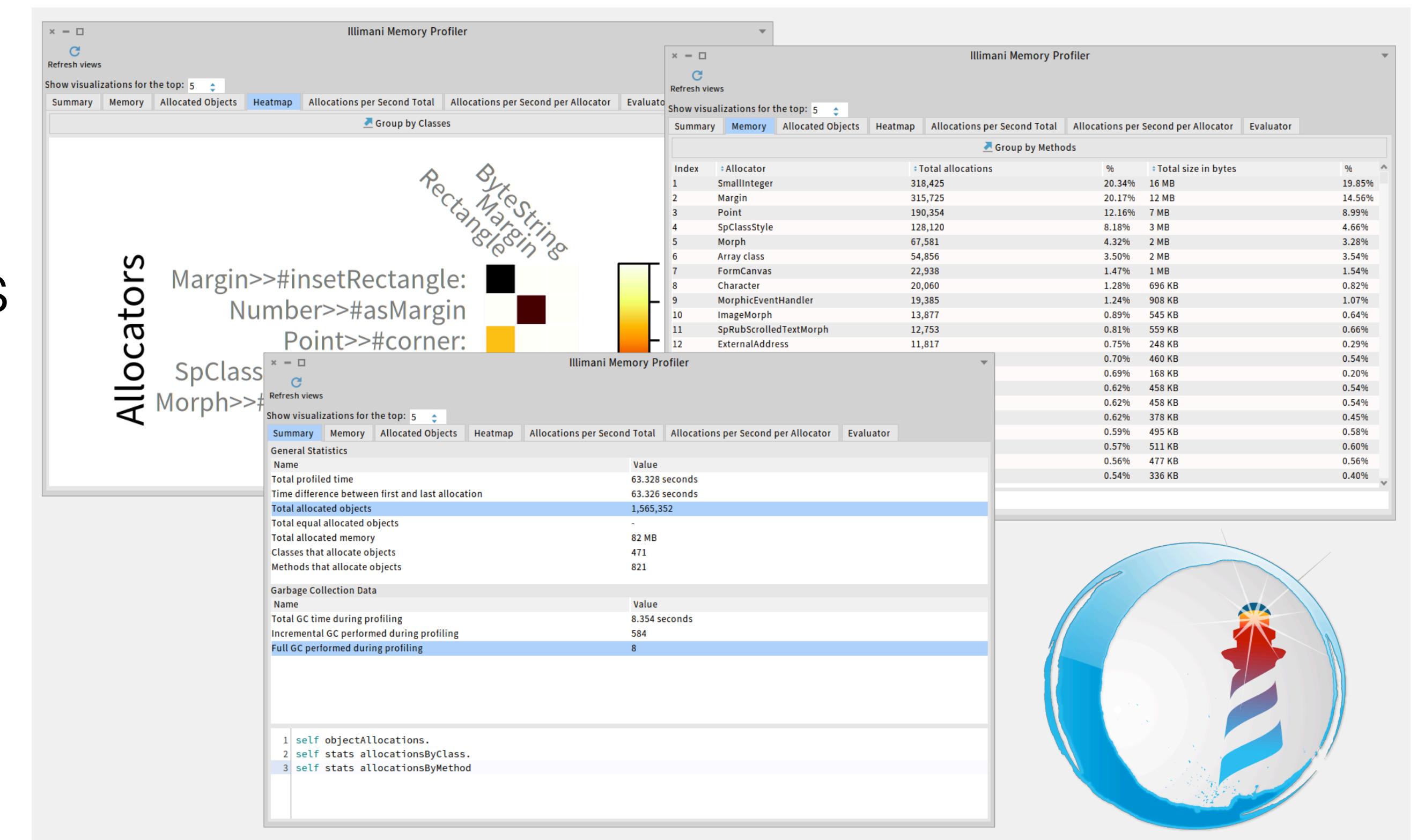
- Identification des sites d'allocations
- Identification des désallocations
- Étude de cas 1 : Morphic Framework
- Étude de cas 2 : DataFrame

# Objectifs

- Comment peut-on écrire des logiciels plus frugaux ?
- Comment peut-on détecter les sites d'allocation d'objets ?
- Est-il possible d'améliorer la performance grâce à des informations sur le profil de durée de vie des objets ?

# Illimani : un Profiler de Mémoire pour Pharo

- License open-source MIT
- Détecter sites d'allocation
- Détecter la durée de vie d'objets
- Matrice d'allocations
- Chart de densité
- Tableaux de mémoire
- Modèle orienté objet riche



[github.com/jordanmontt/illimani-memory-profiler](https://github.com/jordanmontt/illimani-memory-profiler)

# Pourquoi identifier les sites d'allocation d'objets ?

- Aide le développeur avec
  - Informations précises sur l'utilisation de la mémoire
  - Concevoir des refactorings pour réduire l'empreinte de mémoire

# Pourquoi connaitre la durée de vie des objets ?

- Permet faire des **optimisations**
  - Pre-tenuring (Blackburn et al.)
  - GC tuning (Lengauer et al., Yang et al., Brecht et al.)

# Plan

- Identification des sites d'allocations
- Identification des désallocations
- Étude de cas 1 : Morphic Framework
- Étude de cas 2 : DataFrame

# Sites d'allocation d'objets

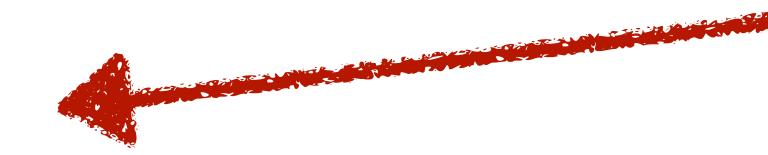
*Un site d'Allocation d'objets est la localisation dans le code source de la creation d'un objet (Clifford et al.)*

```
AthensTextScanner >> initialize
```

```
    lines := OrderedCollection new
```

```
    . . .
```

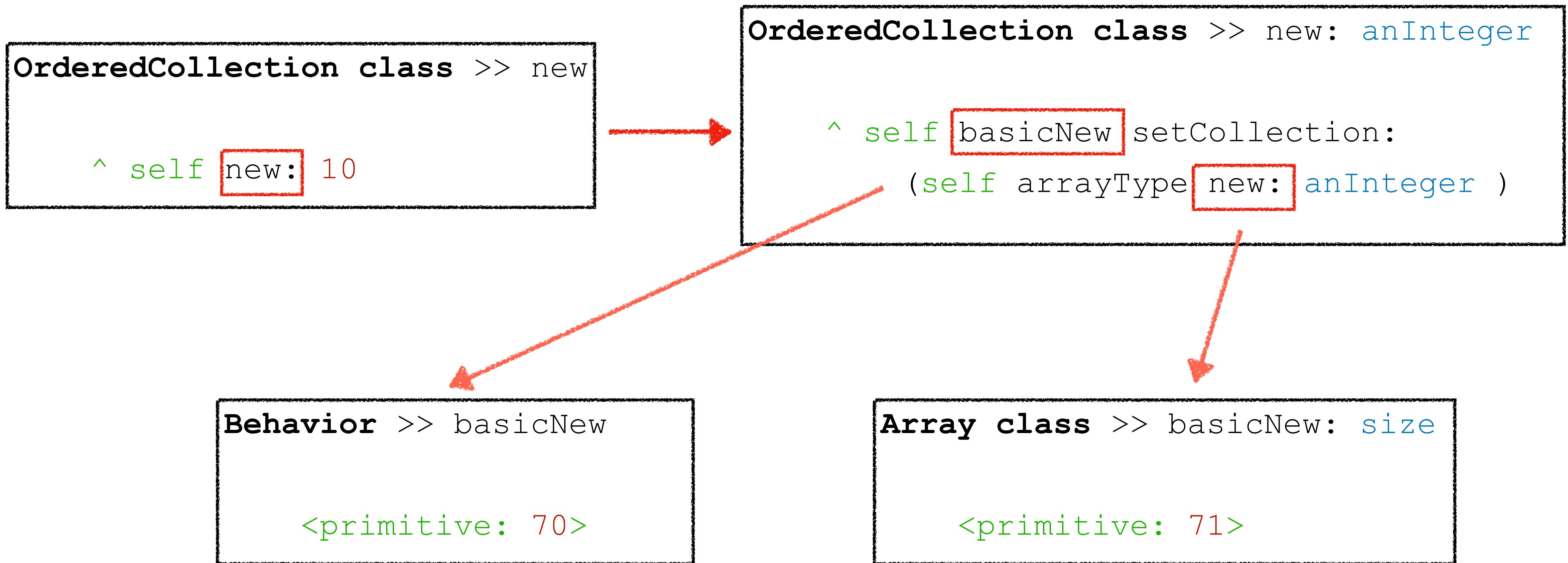
Site d'Allocation



# Capturer les sites d'allocation

En Pharo, presque tous les calculs sont faits par l'envoie des messages (Bergel et al). Cela est aussi le cas pour allouer un objet.

# Capturer les sites d'allocation

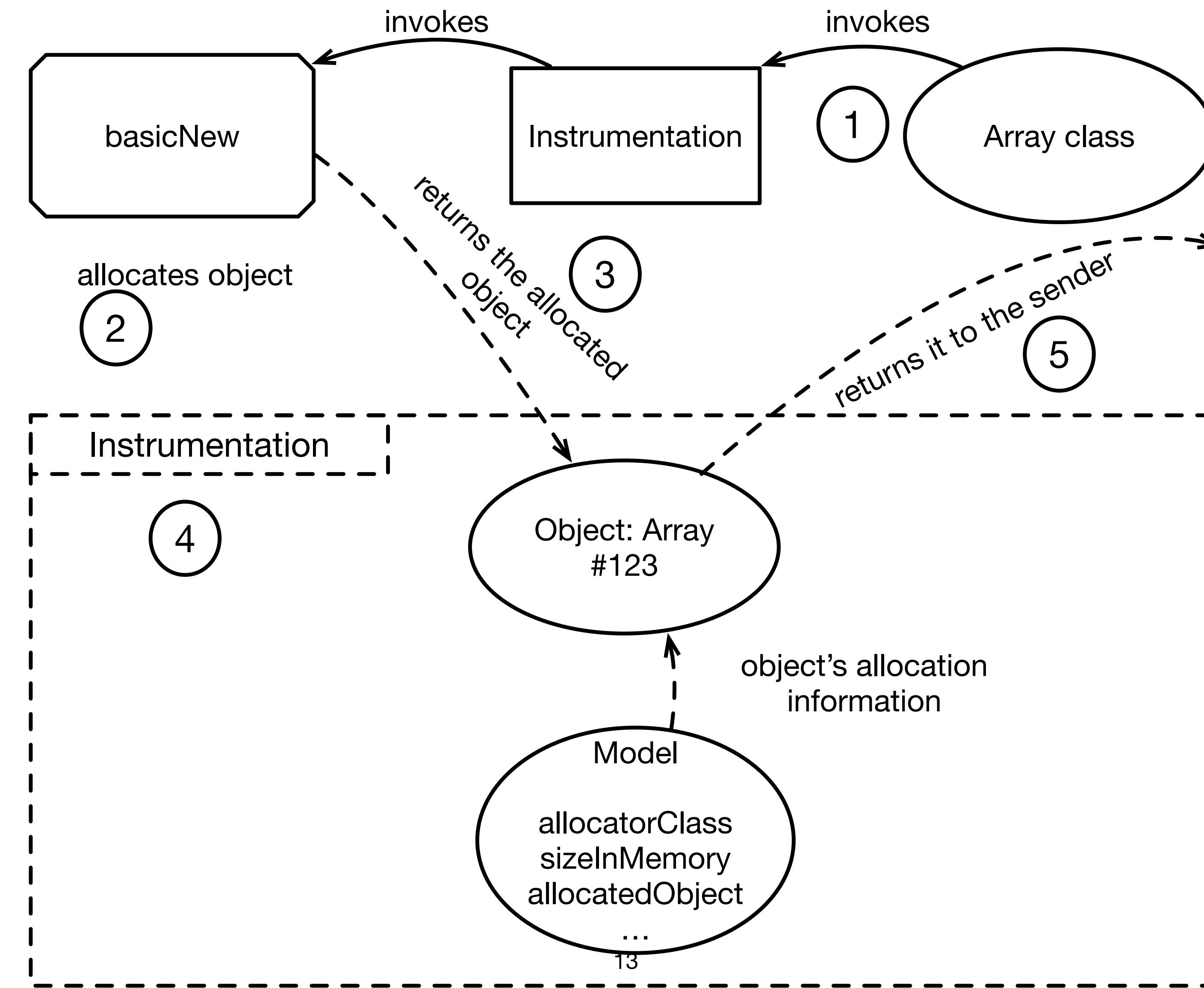


# Sites d'allocation ambiguës en Pharo

Nous avons instrumenté 4 méthodes qui allouent d'objets pour capturer quand elles seront appelées pour après filtrer la pile d'exécution.

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

# Instrumentation via Proxies



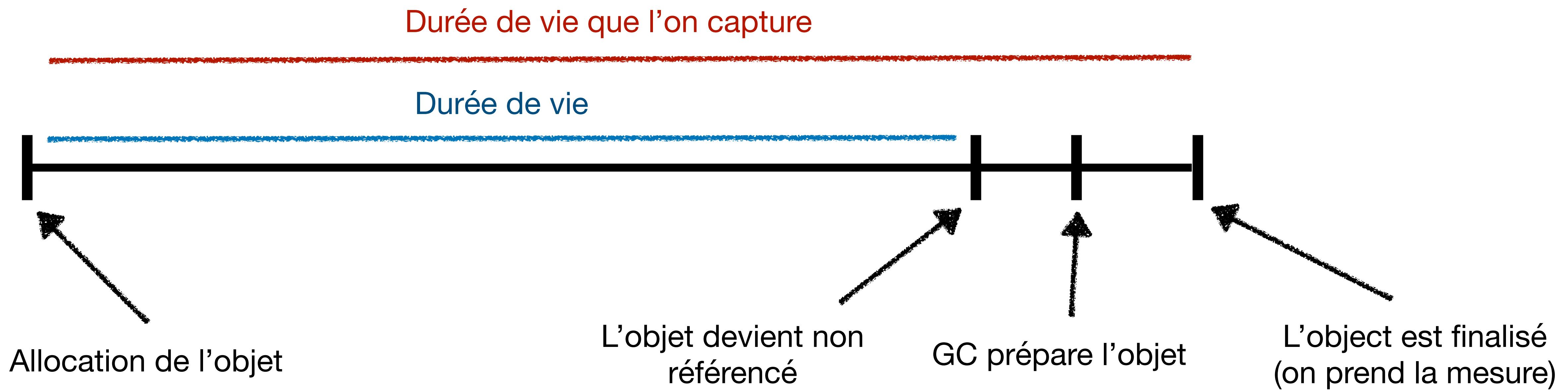
# Plan

- Identification des sites d'allocations
- Identification des désallocations
- Étude de cas 1 : Morphic Framework
- Étude de cas 2 : DataFrame

# Durée de vie d'un objet

$$\text{duréeDeVie} = \text{tempsDeFinalisation} - \text{tempsD'Allocation}$$

# Durée de vie d'un objet

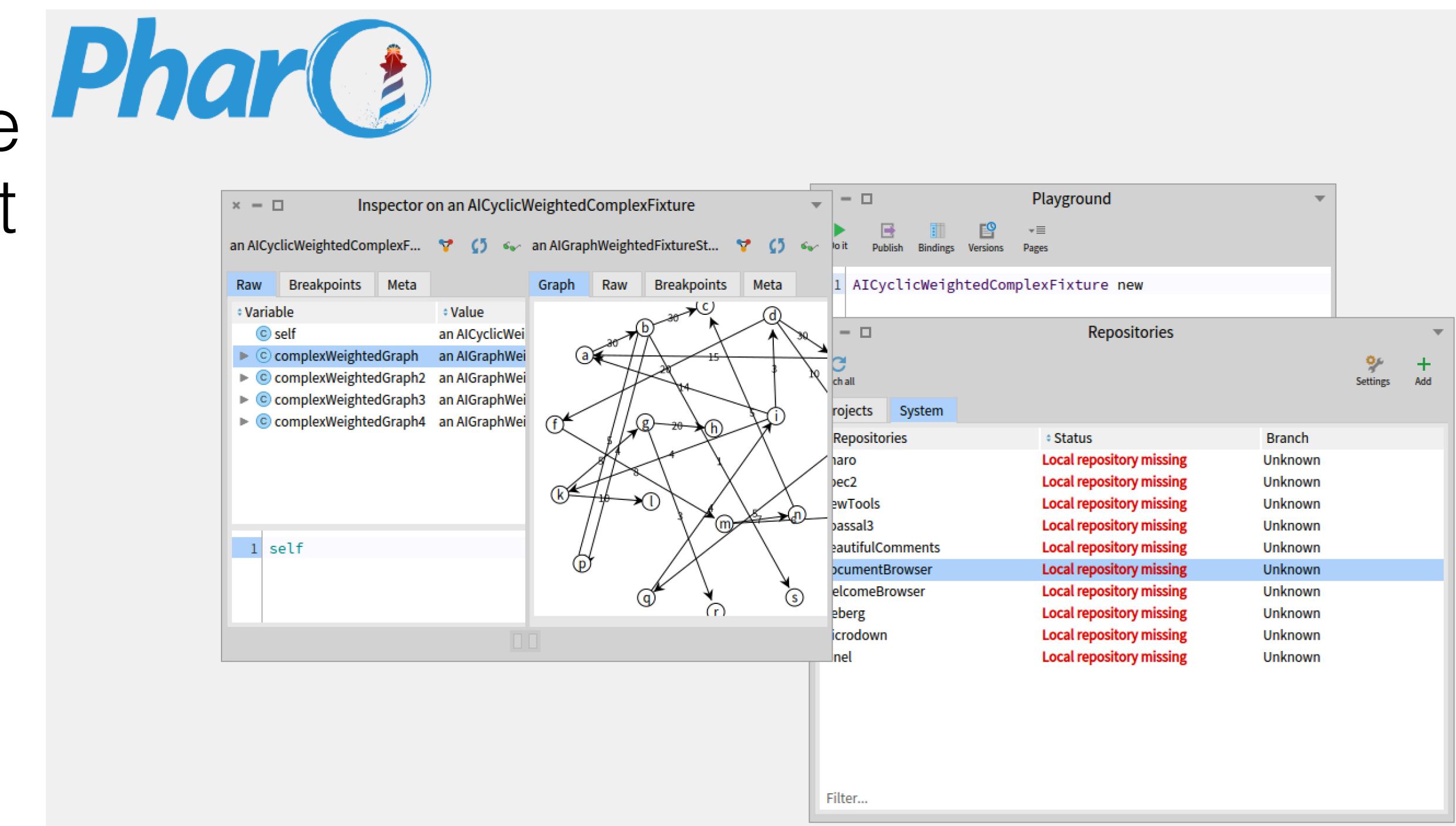


# Plan

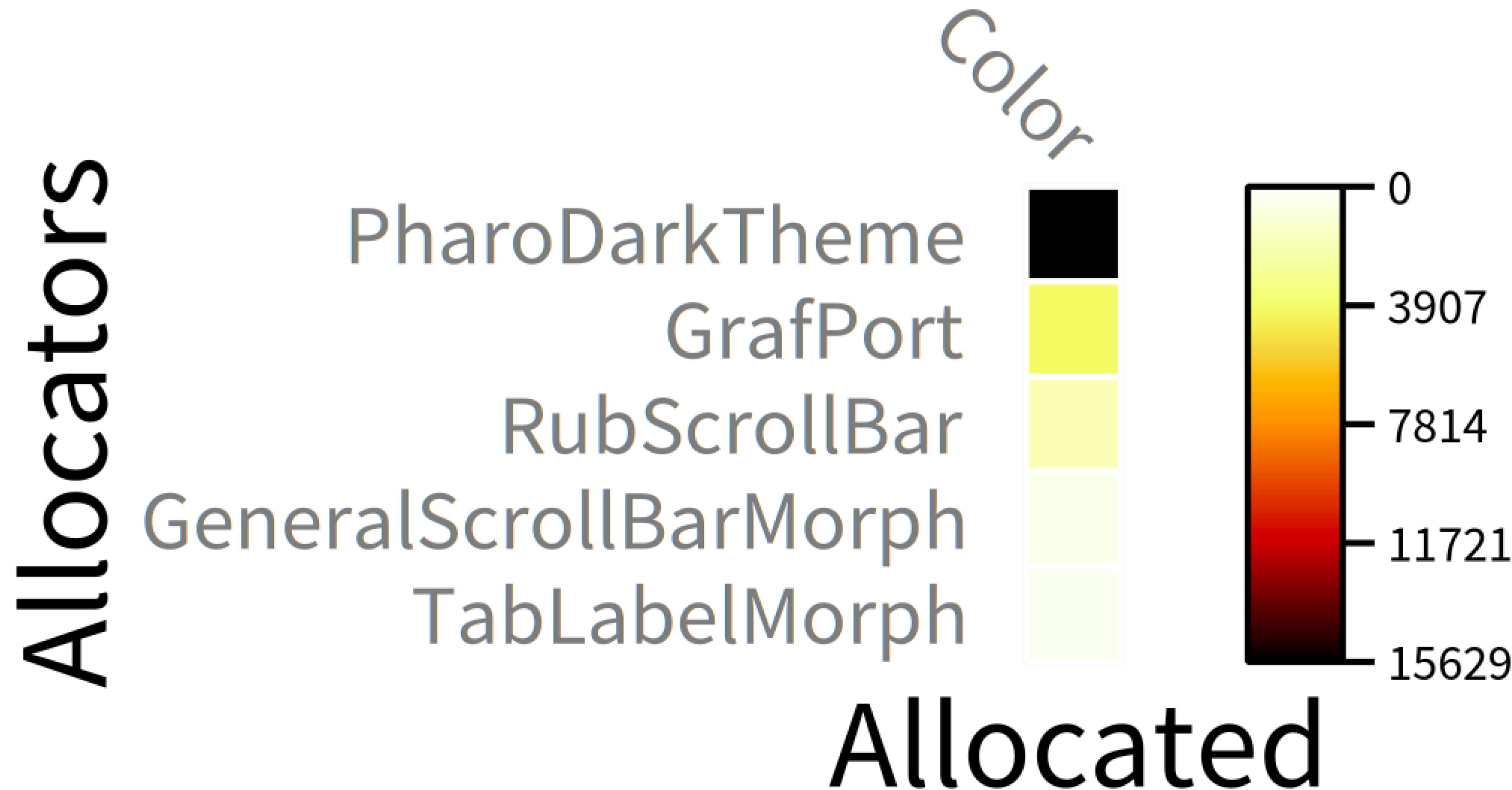
- Identification des sites d'allocations
- Identification des désallocations
- Étude de cas 1 : Morphic Framework
- Étude de cas 2 : DataFrame

# Étude de cas 1 : Morphic UI Framework

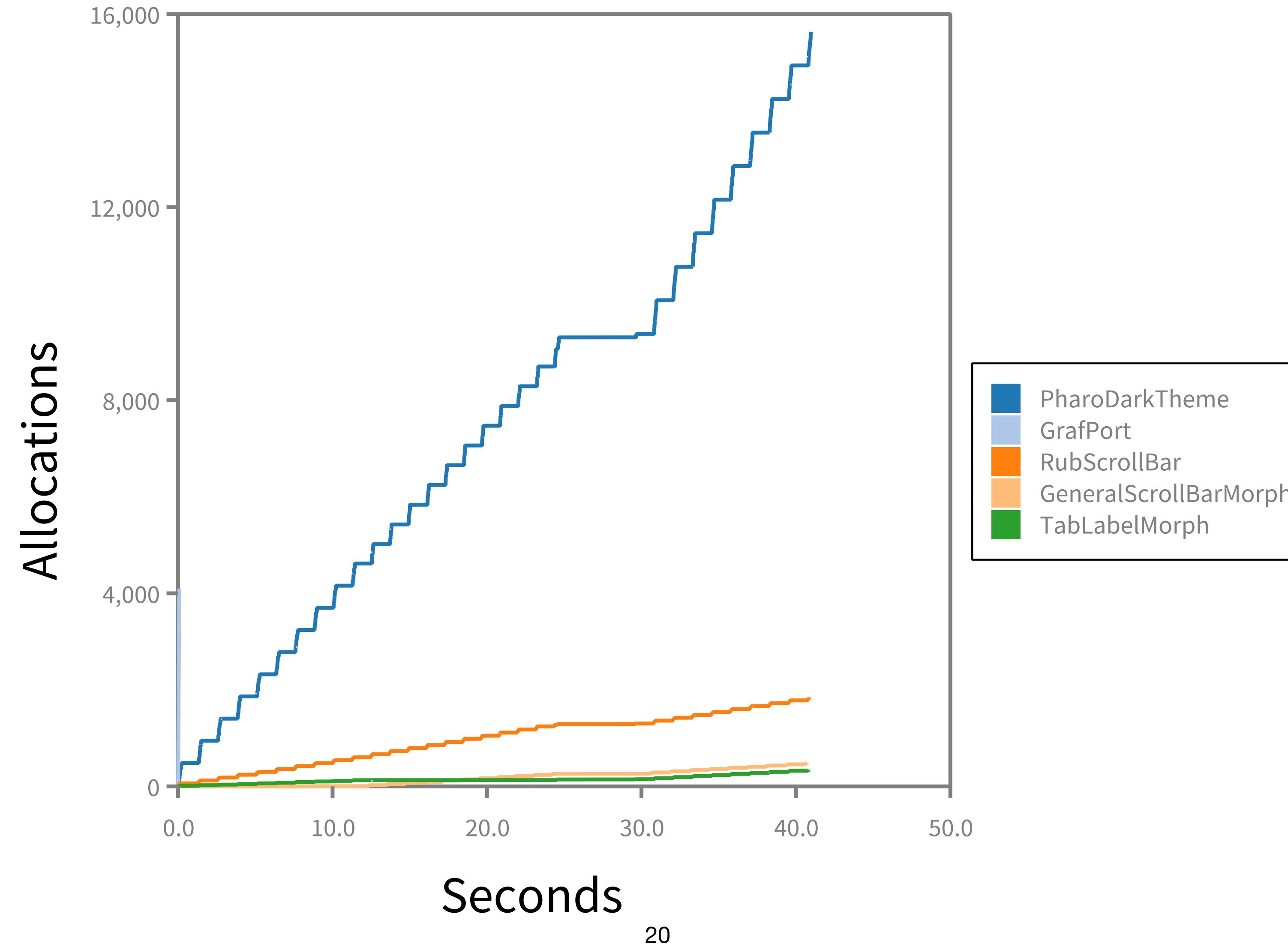
- Profiler des allocations de type **Color**
- Nous avons ouvert 30 outils de l'IDE de Pharo et nous les avons rendu pendant 100 cycles



# Matrice d'allocation



# Allocations dans le temps

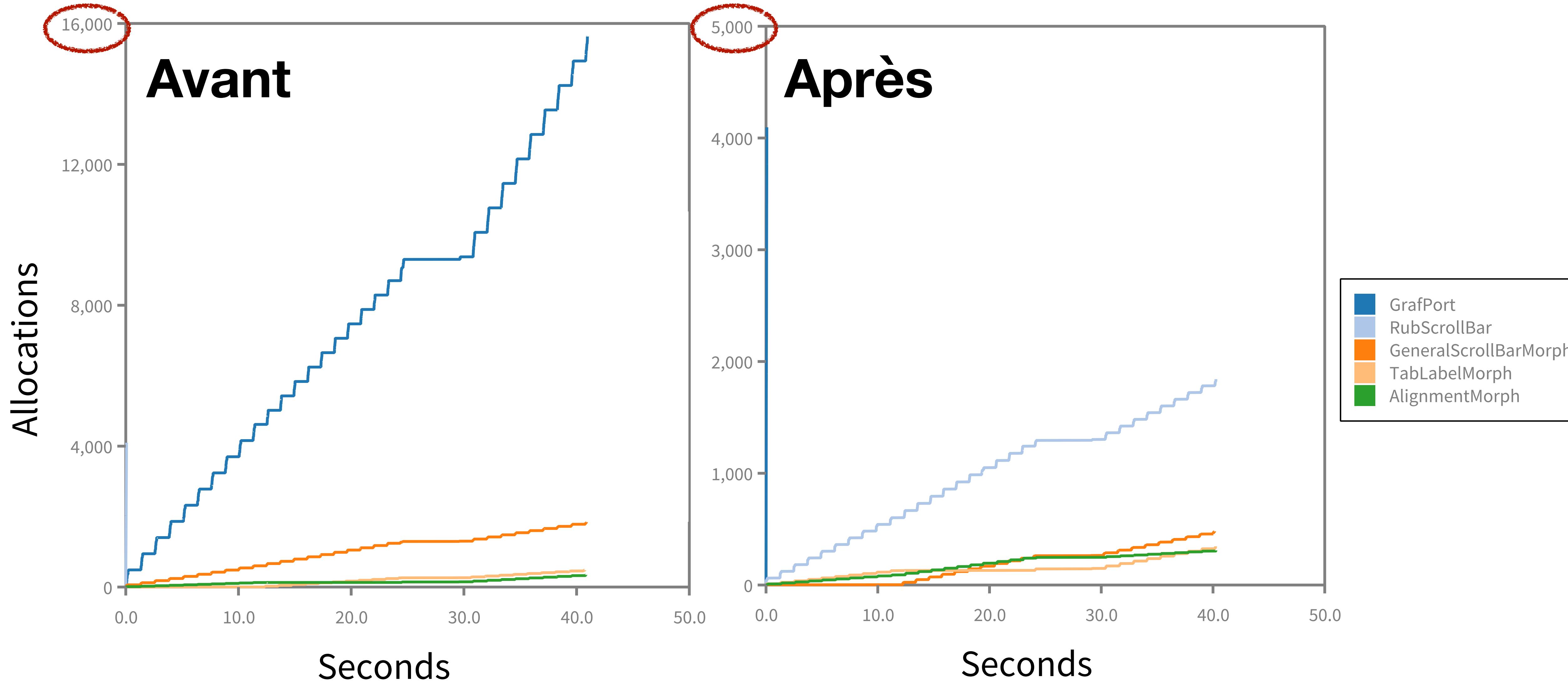


# Analyse Morphic

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%

Nous avons identifié un site d'allocation dans la classe PharoDarkTheme qui allouait 66% de toutes les couleurs avec 99,9% d'allocations redondantes.

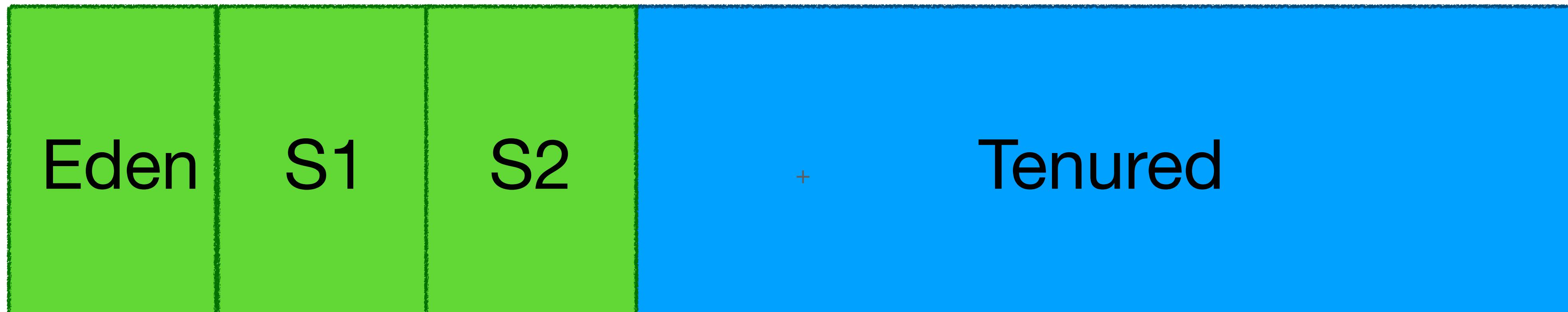
# Morphic après la Correction



# Plan

- Identification des sites d'allocations
- Identification des désallocations
- Étude de cas 1 : Morphic Framework
- Étude de cas 2 : DataFrame

# Gestion de mémoire en Pharo

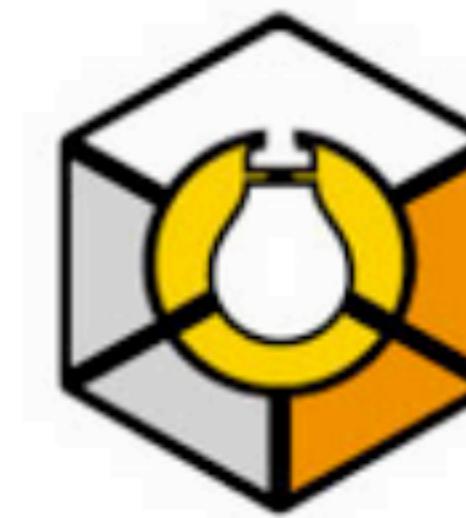


Young generation

Old generation

# Étude de cas 2 : DataFrame

PolyMathOrg/  
**DataFrame**



DataFrame in Pharo - tabular data structures for  
data analysis

12  
Contributors

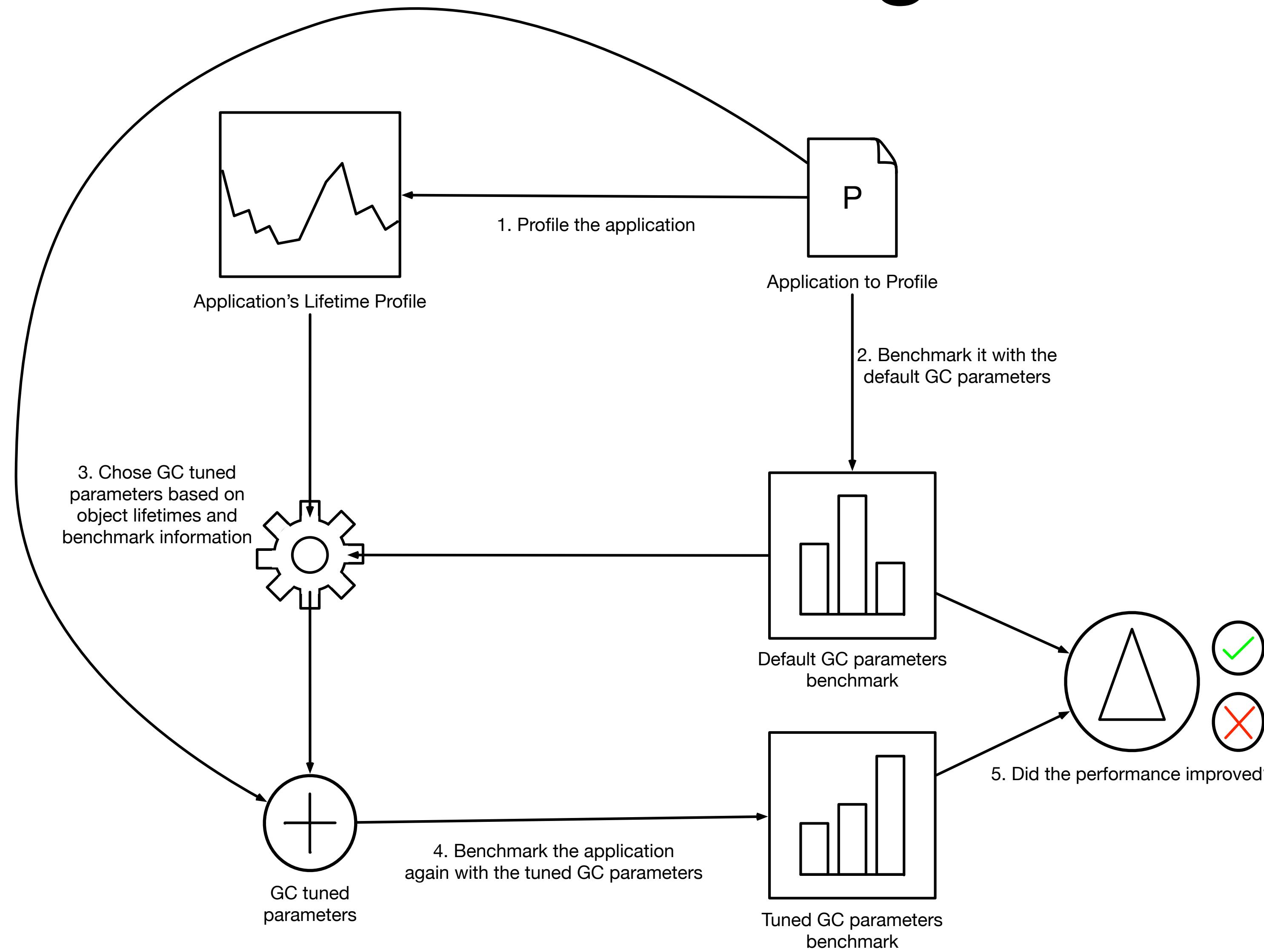
37  
Issues

67  
Stars

21  
Forks



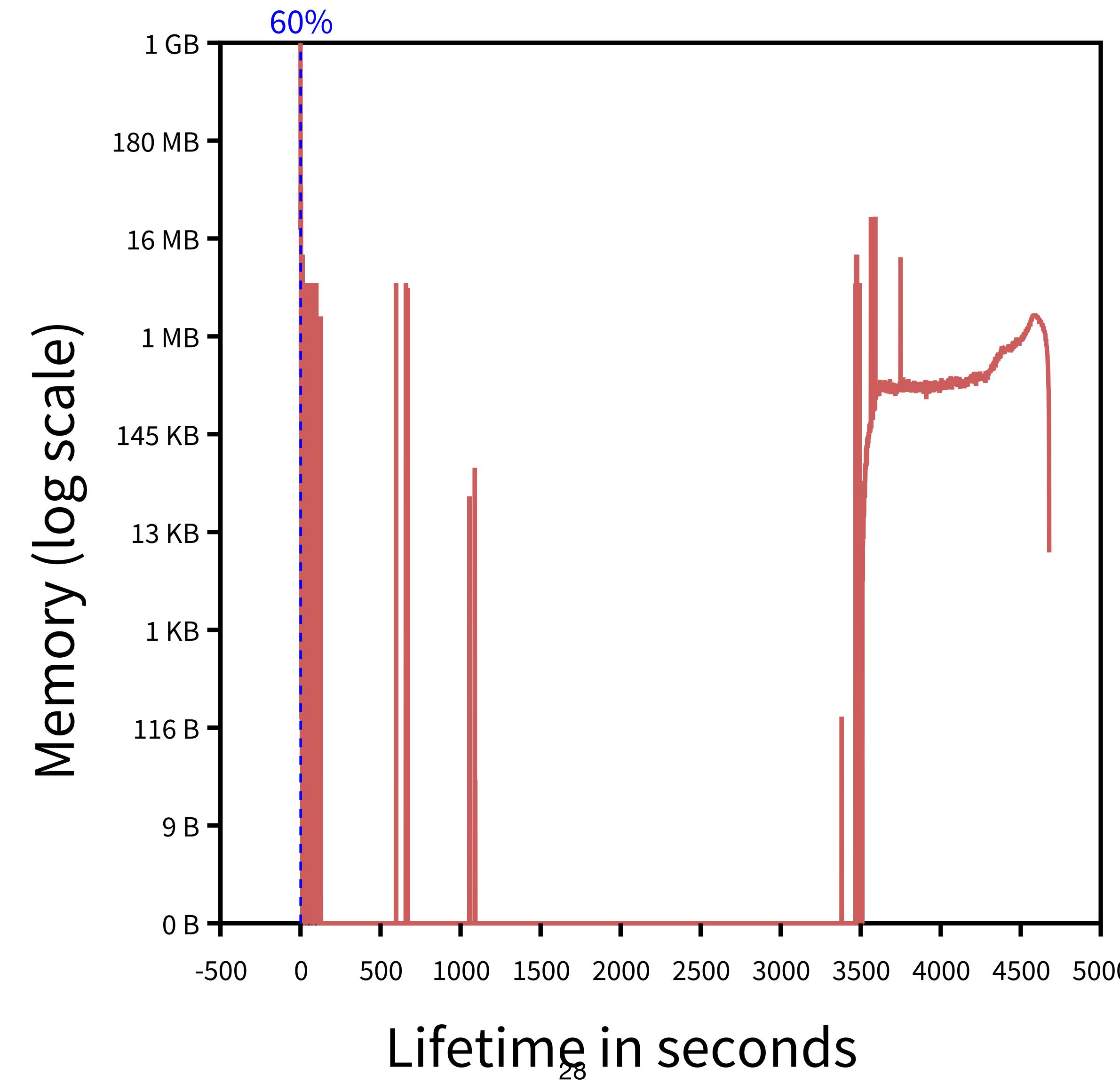
# Étude de cas 2 : Méthodologie



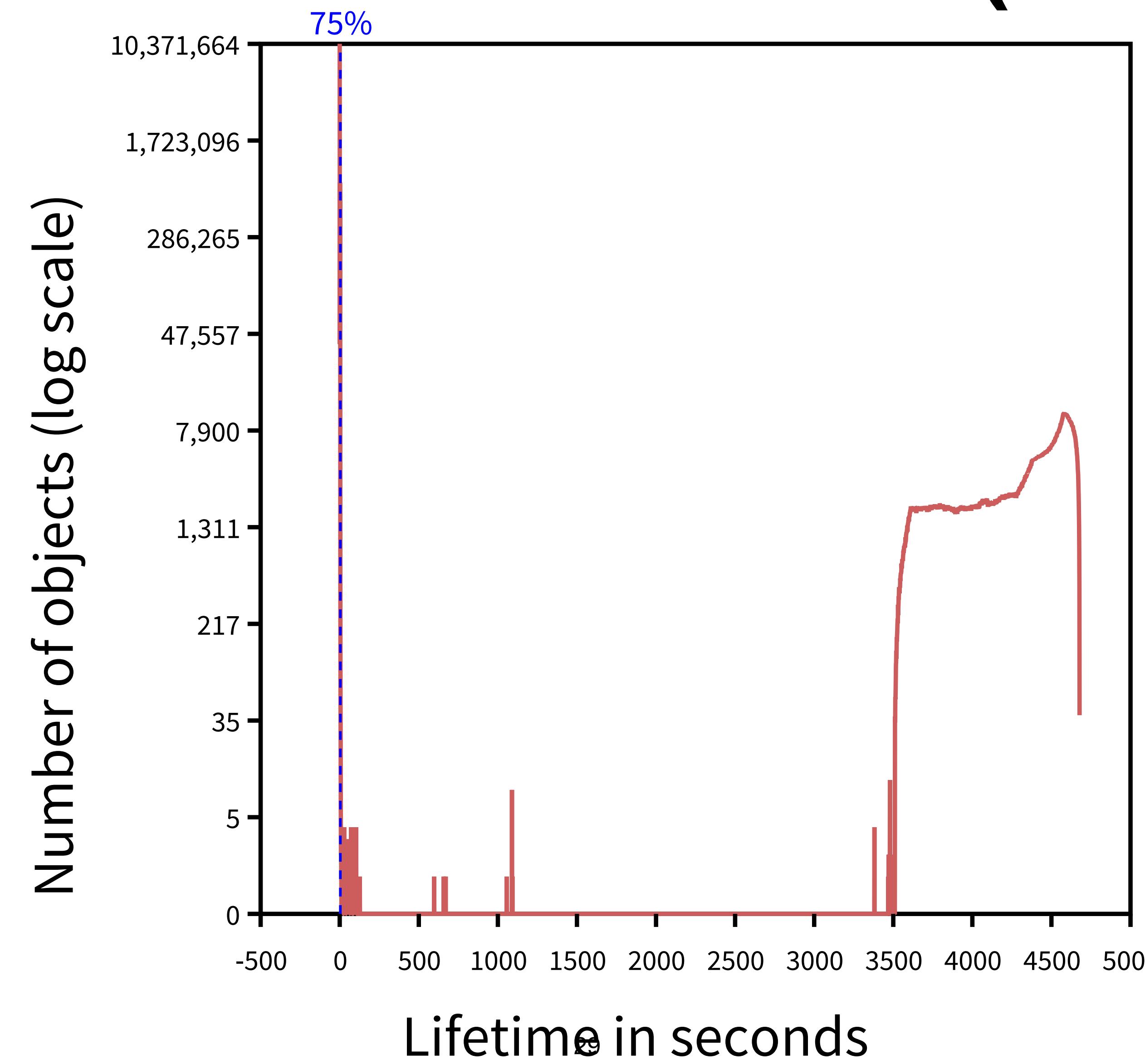
# Benchmarking DataFrame

<b>Dataset</b>	<b># of scavengers</b>	<b># of full GCs</b>	<b>GC time</b>	<b>Total time</b>	<b>GC time in %</b>
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%

# Chart de densité d'allocations (mémoire)



# Chart de densité d'allocations (# objets)



# Améliorations DataFrame

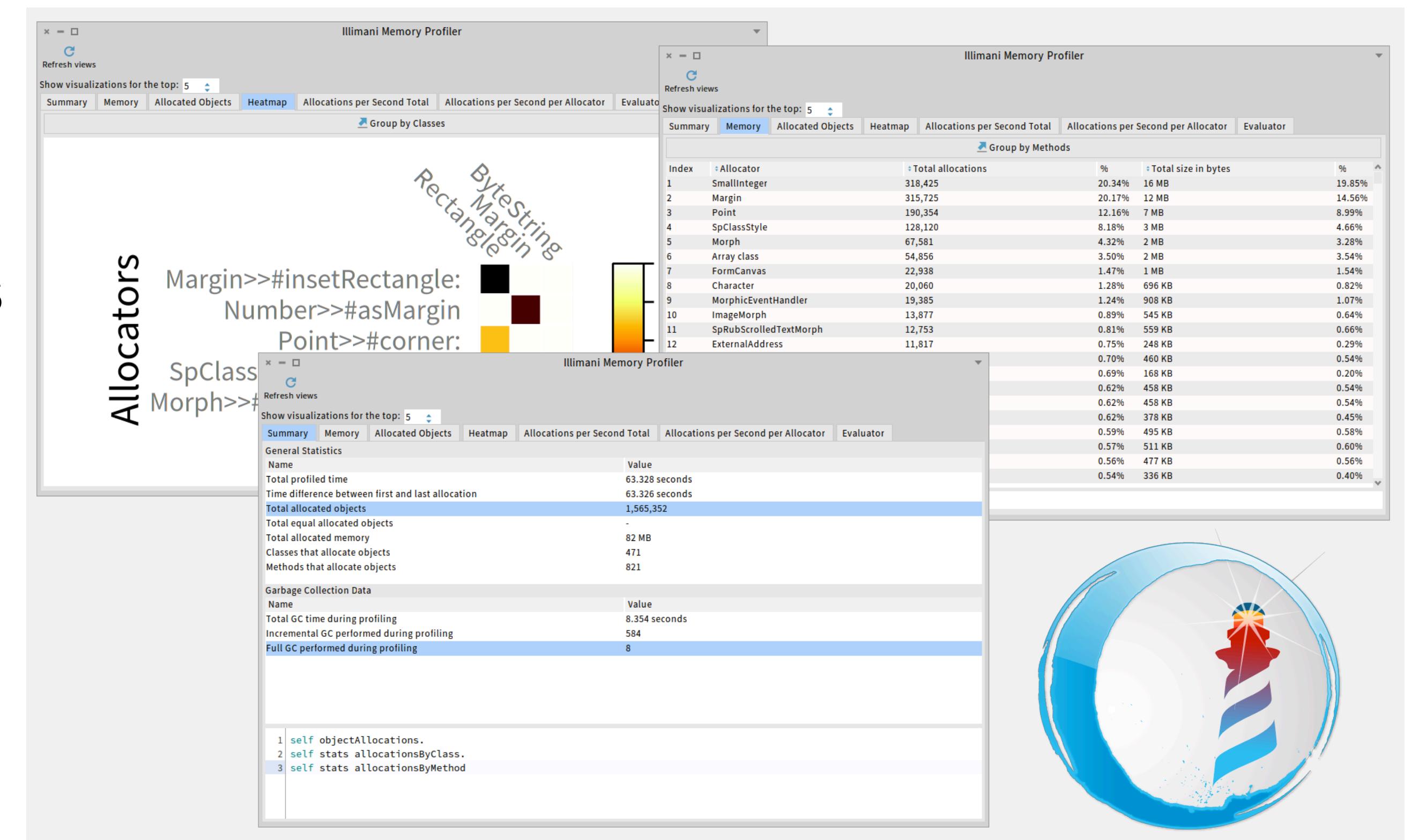
<b>GC Configuration</b>	<b>GC spent time</b>	<b>Total execution time</b>	<b>Improved performance</b>
Default	58 min 18 sec	1 hour 6 min 18 sec	1×
Configuration 1	9 min 41 sec	17 min 46 sec	3.7×
Configuration 2	4 min 57 sec	12 min 54 sec	5.1×
Configuration 3	5 min 8 sec	13 min 2 sec	5.1×
Configuration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

# Resultats

- Illimani, un profiler de mémoire pour Pharo
- Jordan Montaño S., Polito G., Ducasse S., Tesone P. (2023). Illimani Memory Profiler at Work: Identifying Object Allocation Sites, A technical report.
- Jordan Montaño S., Palumbo N., Polito G., Ducasse., Tesone P. (2023). Improving Performance Through Object Lifetimes Profiling: the DataFrame Case. IWST'23, Lyon, France. (accepted).

# Illimani : un Profiler de Mémoire pour Pharo

- License open-source MIT
- Détecter sites d'allocation
- Détecter la durée de vie d'objets
- Matrice d'allocations
- Chart de densité
- Tableaux de mémoire
- Modèle orienté objet riche



[github.com/jordanmontt/illimani-memory-profiler](https://github.com/jordanmontt/illimani-memory-profiler)