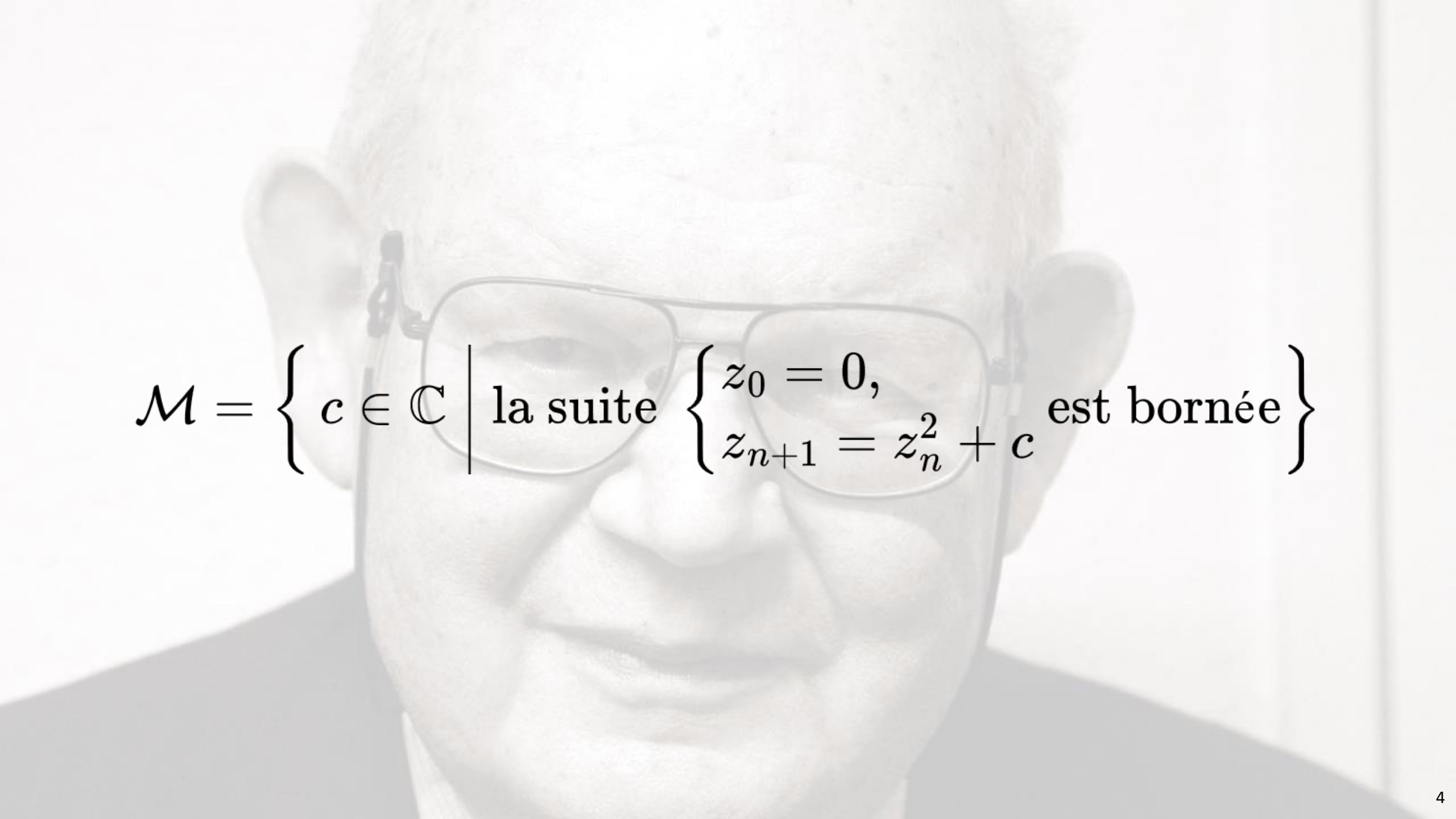
“Les value types ne sont pas ℂomplexes”

Clément de Tastes

{riviera dev}

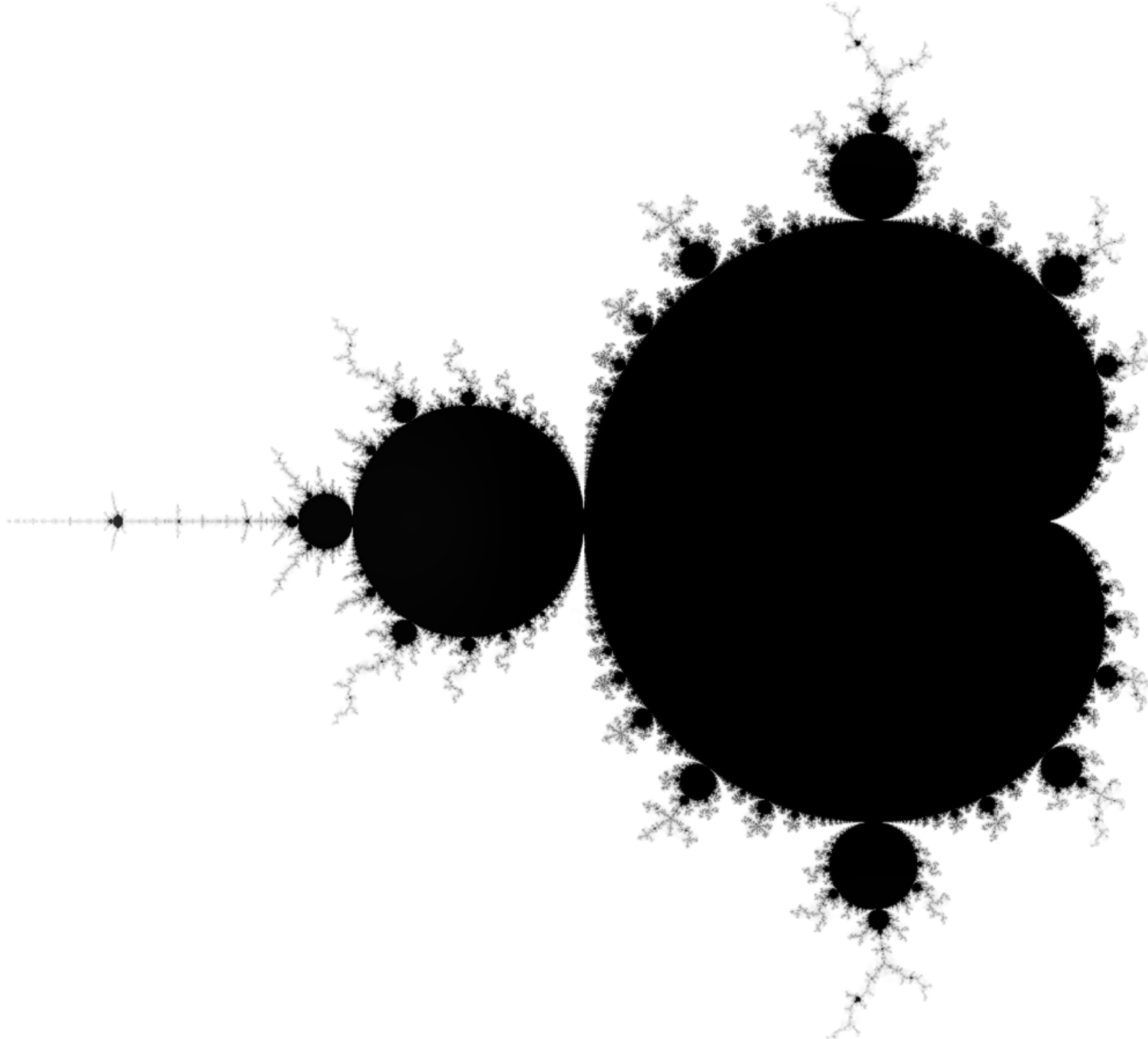
SCIAM

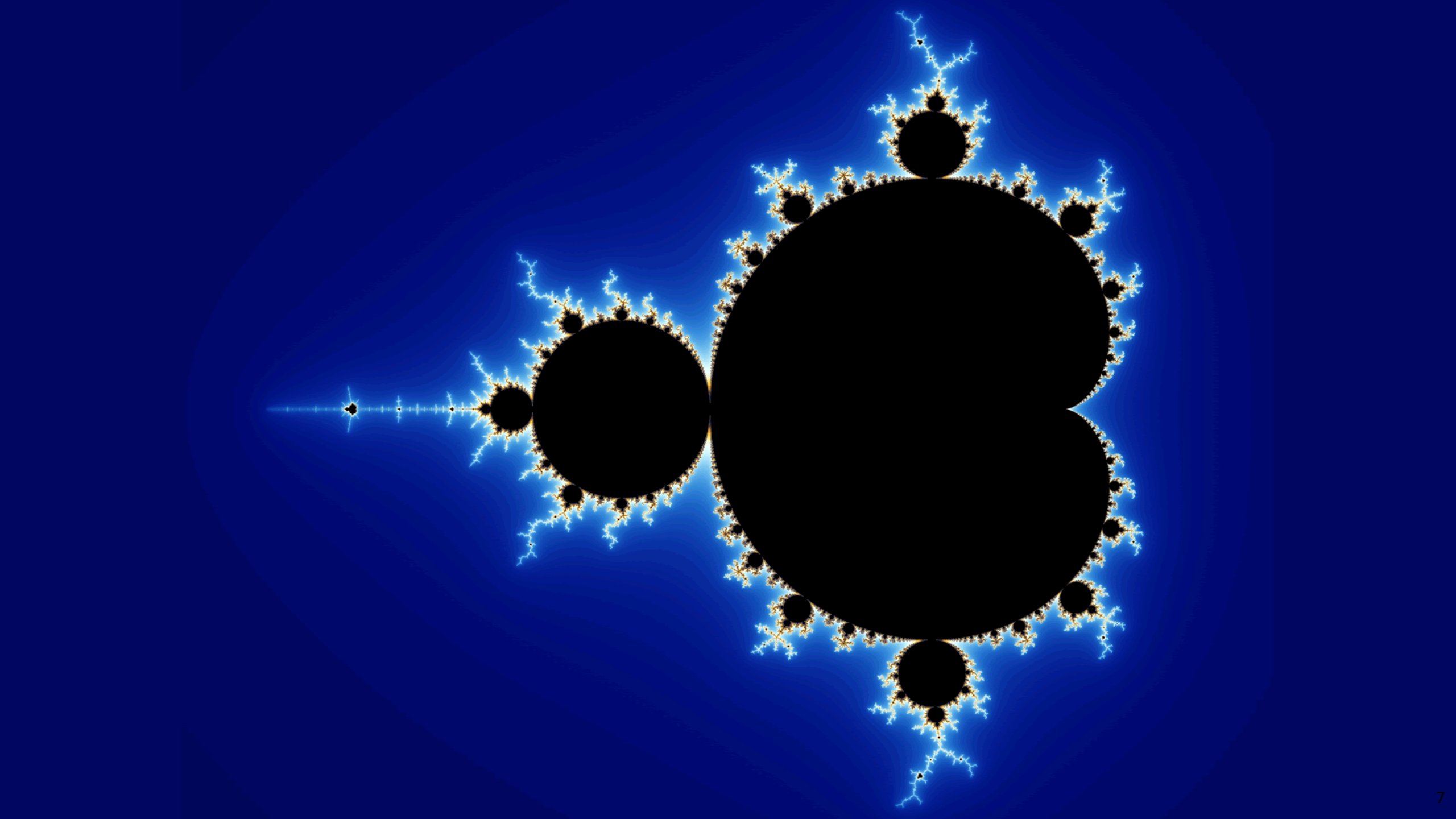


$$\mathcal{M} = \left\{ c \in \mathbb{C} \;\middle|\; \text{la suite } \begin{cases} z_0 = 0, \\ z_{n+1} = z_n^2 + c \end{cases} \text{ est bornée} \right\}$$

Démo

“Escape”





$$z_{n+1} = z_n^2 + c$$

```
public int computeEscape(double re0, double im0, int max) {

  double re = 0;
  double im = 0;
  double x2 = 0;
  double y2 = 0;
  int i = 0;
  double modulusSquared = 0;

  while (modulusSquared <= 4 && i < max) {
    im = 2 * re * im + im0;
    re = x2 - y2 + re0;
    x2 = re * re;
    y2 = im * im;
    modulusSquared = x2 + y2;
    i++;
  }

  return i;
}
```

$$z_{n+1} = z_n^2 + c$$

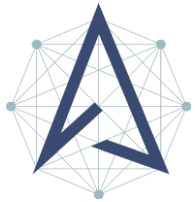
```
public record Complex(double re, double im) {
  // add(), square() ...
}

public int computeEscape(Complex c, int max) {

  Complex z = Complex.ZERO;
  int i = 0;

  while (z.magnitudeSquared() <= 4 && i < max) {
    z = z.square().add(c);
    i++;
  }

  return i;
}
```

Benchmark

1024 x 1024 — itérations max : 255

Benchmark	Mode	Cnt	Score		Error	Units
primitive	avgt	10	**19.758**	±	0.119	ms/op
record	avgt	10	**207.828**	±	1.938	ms/op

Clément de Tastes

- Tech Lead @ SCIAM 

- @cdetastes.bsky.social
- github.com/CodeSimcoe
- blog.sciam.fr
- www.toulon.dev
- Java ☕ — JavaFX — Quarkus

- Je brasse ma bière 🍺

Rémi Forax

- Maître de conférences à l'Université Gustave Eiffel
- github.com/forax
- Specs Java ☕ pour invokedynamic, lambda, module, record, pattern-matching, value-type
- Je maintiens plusieurs projets open-source que personne n'utilise

Disclaimer

Ne croyez pas tout ce que je vais dire

Projet Valhalla

- Projet de l'OpenJDK
- Démarré en 2014 (!!)

"Java's epic refactor"

Brian Goetz

- Enrichir le langage avec des ***value objects***

"Codes like a class, works like an int"

- Combiner abstraction de la POO avec les performances des types primitifs
- Combler le "fossé" entre les types primitifs et les objets

JEP 401: Value Classes and Objects (Preview)

Owner	Dan Smith
Type	Feature
Scope	SE
Status	Submitted
Component	specification
Discussion	valhalla dash dev at openjdk dot org
Effort	XL
Duration	XL
Reviewed by	Alex Buckley, Brian Goetz
Created	2020/08/13 19:31
Updated	2025/11/19 20:33
Issue	8251554

Summary

Enhance the Java Platform with *value objects*: class instances that have only `final` fields and lack object identity. This is a preview language and VM feature.

Goals

- Allow developers to opt in to a programming model for domain values in which objects are distinguished solely by the values of their fields, much as the `int` value 3 is distinguished from the `int` value 4.
- Support compatible migration of existing classes that represent domain values to this programming model. Migrate suitable classes in the JDK, such as `Integer` and `LocalDate`, to be value classes.
- Maximize the freedom of the JVM to store domain values in ways that improve memory footprint, locality, and garbage collection efficiency.

Stack (pile)

Utilisée pour l'execution des méthodes

Chaque appel de méthode pousse une nouvelle frame sur la pile

Stocke les variables locales pour l'interpréteur (par multiple de 32 bits)

Heap (tas)

Utilisé pour le stockage des objets

Chaque **new** alloue l'espace mémoire pour l'objet et son header

Stocke les champs (par multiple de 8 bits)

Les types primitifs sont stockés directement
Les références aux objets sont stockées sous forme de pointeurs vers la heap

Classes	Value Classes	Primitifs
Identité	Pas d'identité	Pas d'identité
Champs, méthodes et peuvent implementer des interfaces Encapsulation	Champs, méthodes et peuvent implementer des interfaces Encapsulation	
	Aplaties sur la stack Peuvent être aplaties sur la heap	Aplatis sur la stack & la heap
Instances nullables	Instances peuvent être nullables	
Intégrité	Peuvent ne pas avoir d'intégrité	Peuvent ne pas avoir d'intégrité (long/double CPU 32bits)*

**JLS 17.7. Non-Atomic Treatment of double and long*

Nouveau mot clé : `value`

Permet de définir une **`value class`**
ou un `value record`

```
public value record Complex(double re, double im) {
  // ...
}
```

Benchmark

1024 x 1024 — itérations max : 255

Benchmark	Mode	Cnt	Score		Error	Units
primitive	avgt	10	**19.758**	±	0.119	ms/op
record	avgt	10	**207.828**	±	1.938	ms/op
value record	avgt	25	**20.887**	±	0.079	ms/op

Démo

01. MandelbrotFX

Java Flight Recorder

Même scenario d’exploration, 3 algorithmes

- Records
- Primitifs
- Value Records

Démo

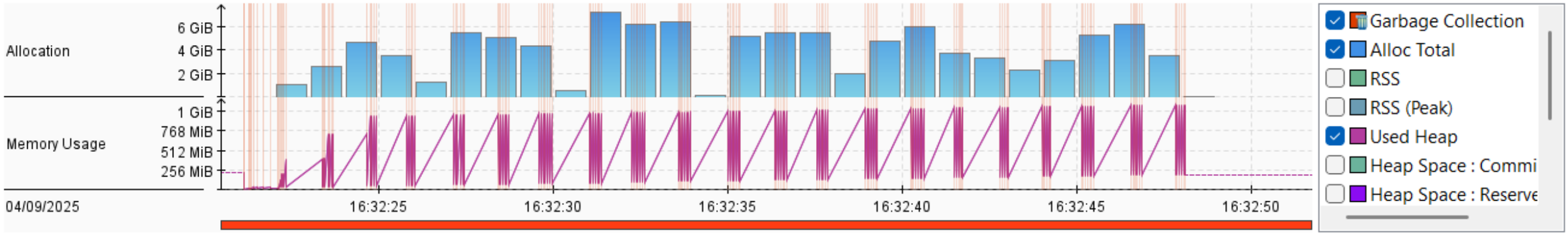
02. Java Mission Control

Records

Memory

Focus: <No Selection> Aspect: <No Selection> Show concurrent: Contained Same threads Time Range: Set Clear

Class	Max Live Count	Max Live Size	Live Size Incre...	Alloc Total	Total Allocatio...
com.codesimcoe.mandelbrotfx.Complex				107 GiB	99,1 %
java.util.stream.Streams$RangeIntSpliterator				600 MiB	0,544 %
byte[]				309 MiB	0,28 %
char[]				21,3 MiB	0,0193 %
int[]				14,3 MiB	0,0129 %
java.lang.classfile.constantpool.PoolEntry[]				11,3 MiB	0,0102 %
java.lang.String				4,64 MiB	0,0042 %

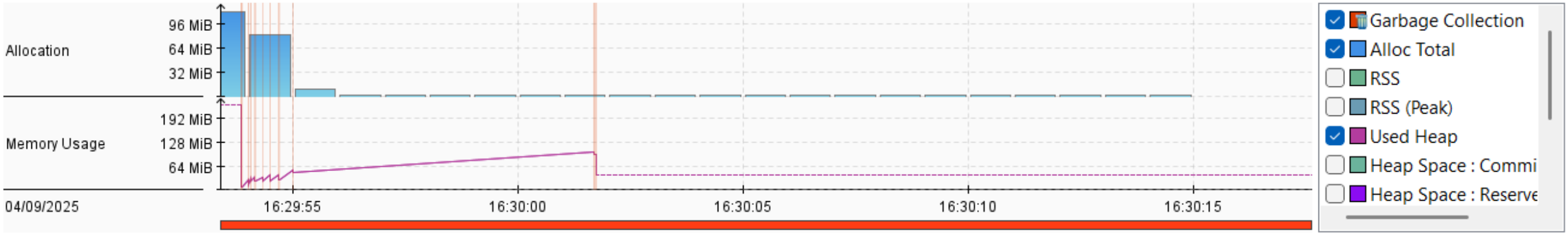


Primitifs

Memory

Focus: <No Selection> Aspect: <No Selection> Show concurrent: Contained Same threads Time Range: Set Clear

Class	Max Live Count	Max Live Size	Live Size Incre...	Alloc Total	Total Allocatio...
byte[]				151 MiB	52,1 %
char[]				76,1 MiB	26,2 %
int[]				13,5 MiB	4,66 %
java.lang.Double				7,84 MiB	2,7 %
java.lang.String				5,89 MiB	2,03 %
java.lang.Class				4,69 MiB	1,62 %
java.io.FileCleanable				2,55 MiB	0,879 %

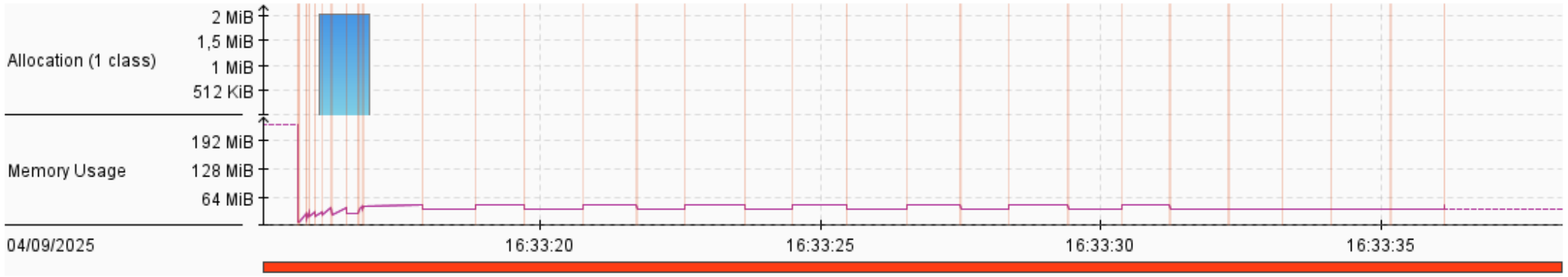


Value Records

Memory

Focus: <No Selection> Aspect: <No Selection> Show concurrent: Contained Same threads Time Range: Set Clear

Class	Max Live Count	Max Live Size	Live Size Incre...	Alloc Total	Total Allocatio...
com.codesimcoe.mandelbrotfx.ValueComplex				2,06 MiB	0,695 %
javafx.css.Size				1,87 MiB	0,631 %
com.sun.prism.d3d.D3DGraphics				1,53 MiB	0,518 %
long[]				1,14 MiB	0,386 %
jdk.internal.classfile.impl.AbstractPoolEntry$Utf8EntryImpl				983 KiB	0,324 %
com.sun.javafx.css.FixedCapacitySet$Single$1				955 KiB	0,315 %
jdk.internal.util.SoftReferenceKey				950 KiB	0,313 %



Ce n’est pas “juste” une optimisation de performances

Démo

03. Propriétés des *value objects*

Initialisation stricte

Les champs doivent être initialisés avant l'appel à `this` ou `super`

JEP 513 : Flexible Constructor Bodies

Démo

04. Initialisation stricte

Changements dans le JDK

- `@ValueBased` -> **Value Types**

- `java.lang : Integer, Long, Float, Double …`
- `java.util : Optional, OptionalInt, OptionalLong …`
- `java.time : LocalDate, LocalTime, Duration, Instant …`

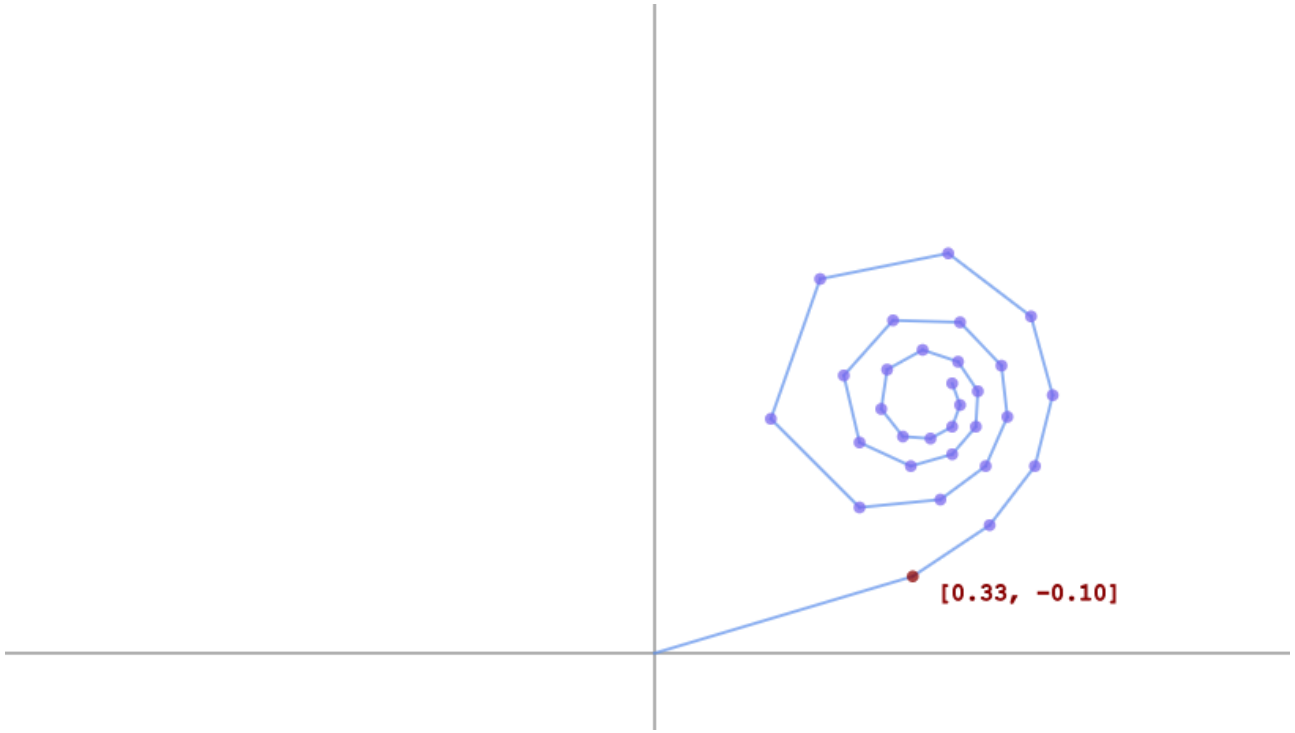
- JEP 390: Warnings for Value-Based Classes *(JDK 16)*

- Pas besoin de recompiler le code externe qui utilise des value types

Démo

05. Wrappers

Orbites

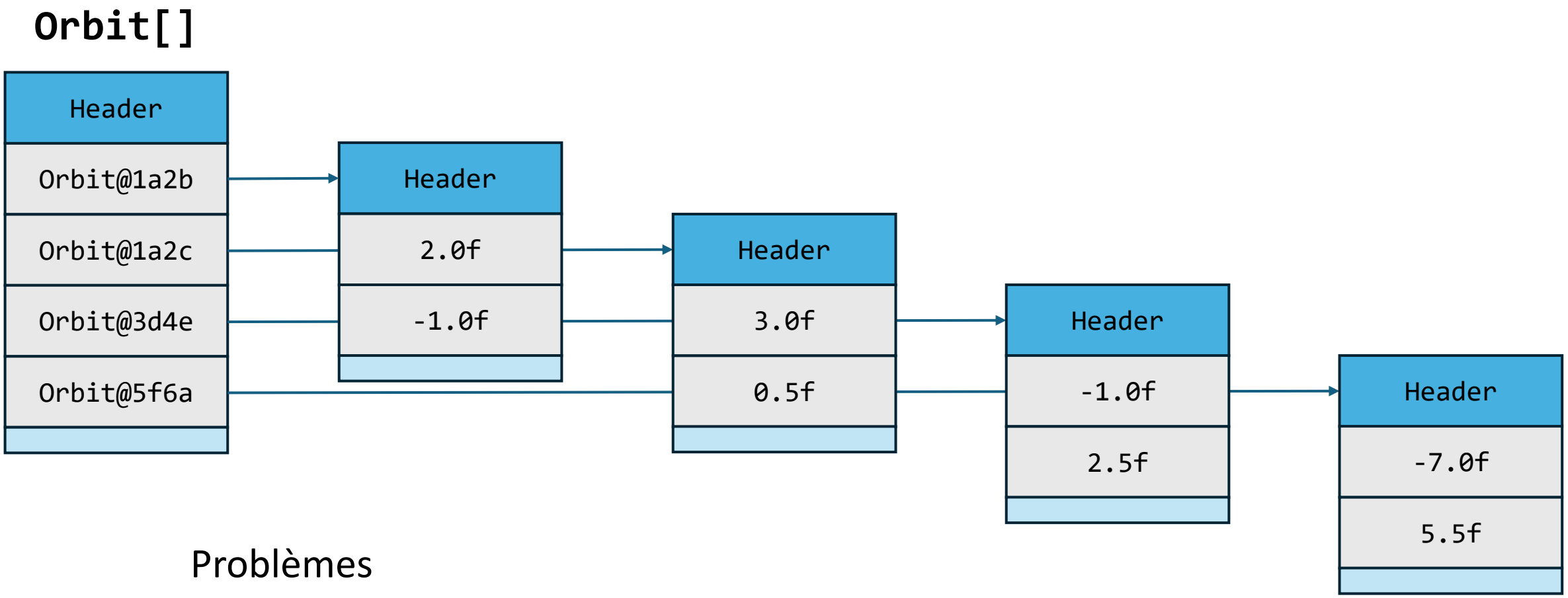


Calcul pour affichage

```
record Orbit(float re, float im) {}
```

Stockés dans un tableau `Orbit[]`

Structure mémoire

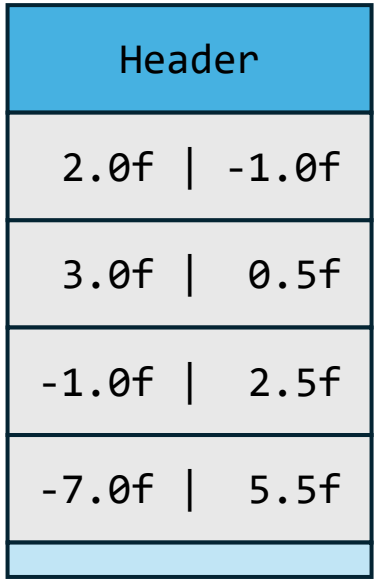


Problèmes

- Surcoût mémoire *(header + padding)*
- Indirections *(pointer chasing)*

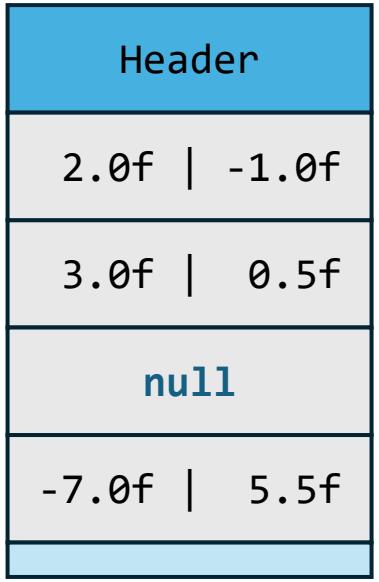
Aplatissement *(flattening) ?* - Tableaux

`ValueOrbit[]`



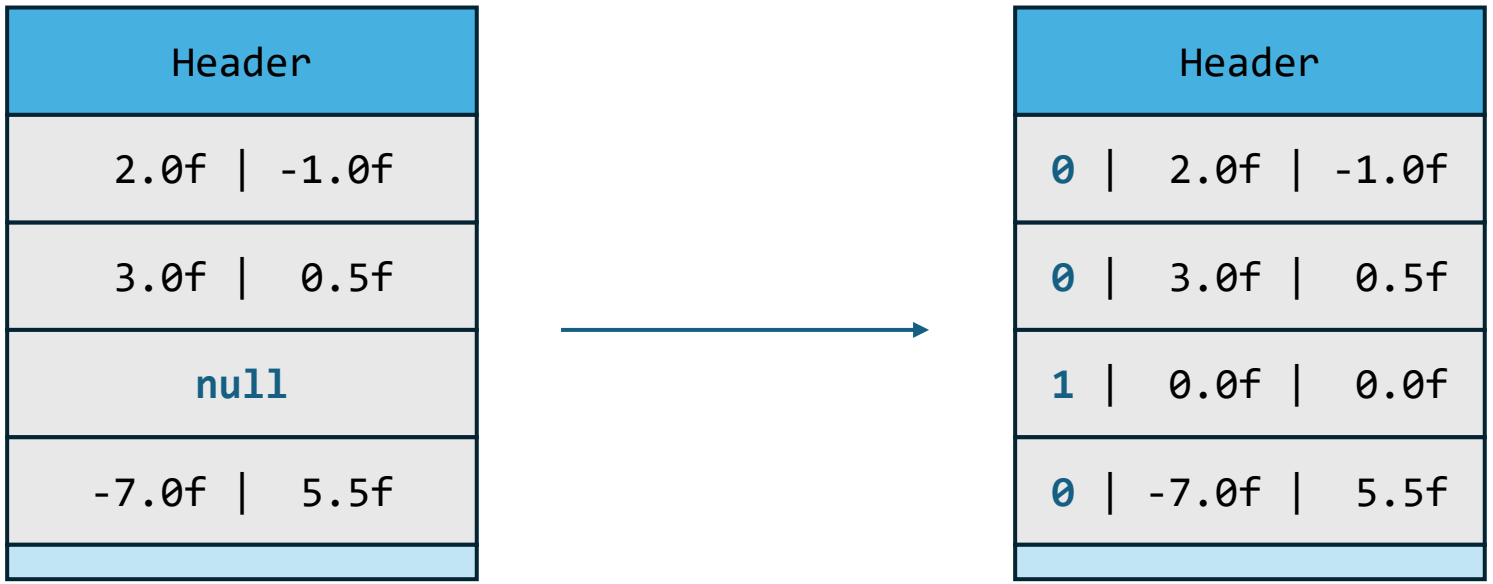
Aplatissement - Tableaux

`ValueOrbit[]`



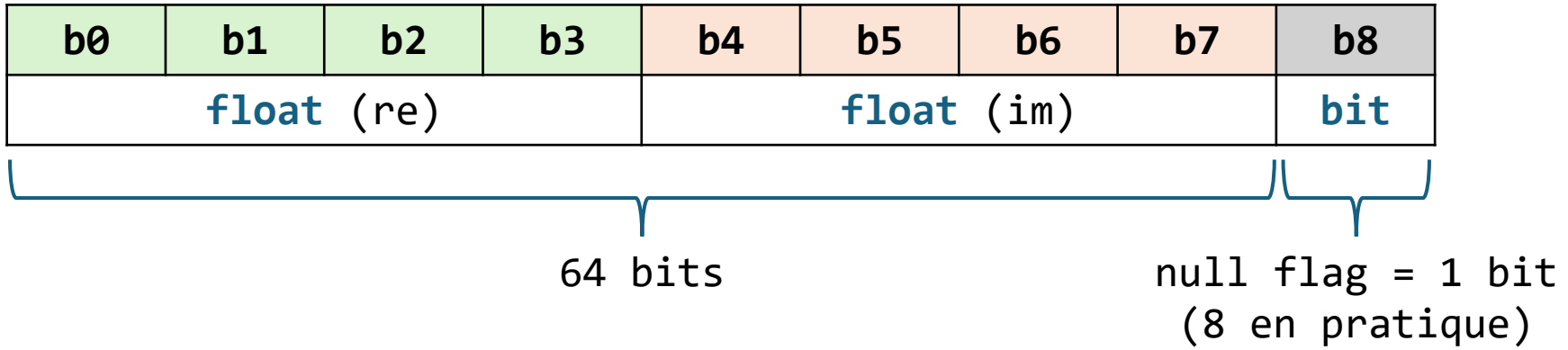
Aplatissement - Tableaux

`ValueOrbit[]`



Aplatissement - Tableaux

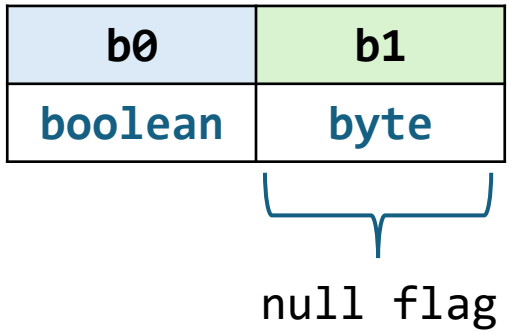
`ValueOrbit[]`



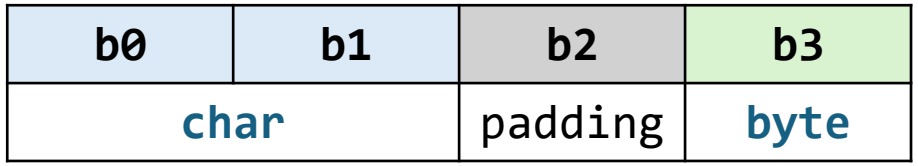
65 bits vs processeur 64 bits

Aplatissement - Tableaux

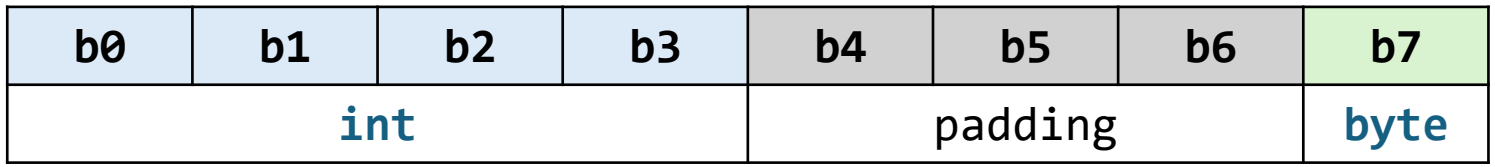
Boolean[]



Character[]



Integer[]



Null-restricted types

JDK-8303099 *: Null-Restricted and Nullable Types*

Définir qu'un type n'accepte pas `null`
A la compilation et au runtime

Syntaxe préssentie : `!` (bang)

JDK-8316779 *: Null-Restricted Value Class Types*

Optimisations pour les value types

Null-restricted types

```
value record ValueOrbit(float x, float y) {}
```

`ValueOrbit!``[]`

Header
2.0f | -1.0f
3.0f | 0.5f
-1.0f | 2.5f
-7.0f | 5.5f

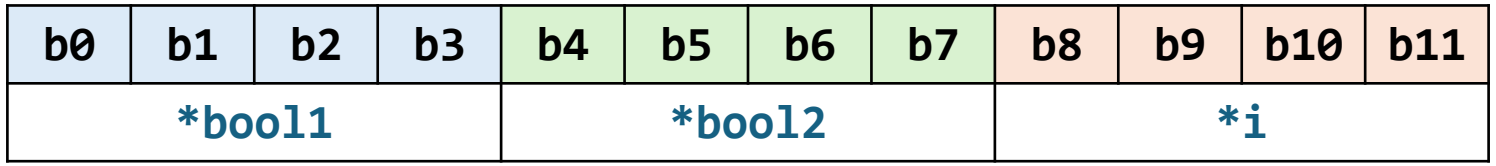
`float+float[]`

Header
2.0f | -1.0f
3.0f | 0.5f
-1.0f | 2.5f
-7.0f | 5.5f

Aplatissement - Champs

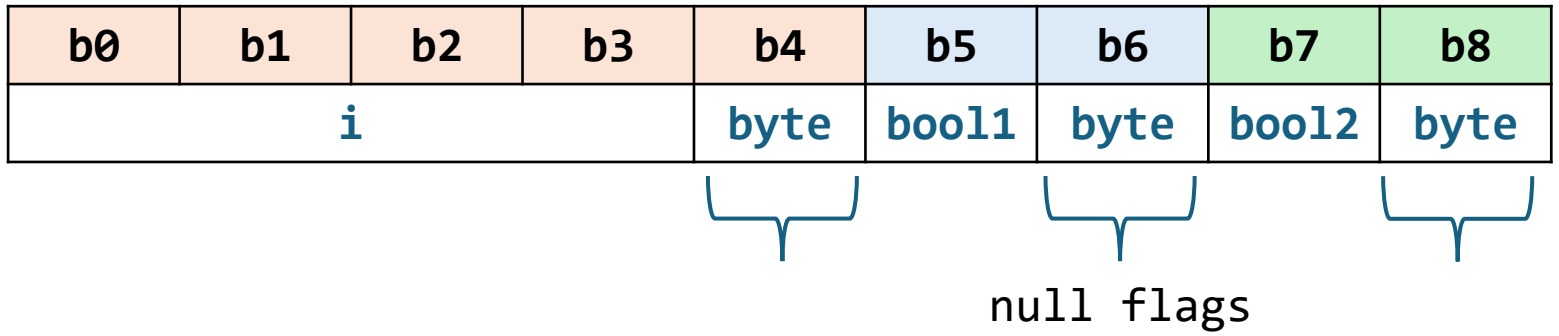
Sans Valhalla

```
record Data(Boolean bool1, Boolean bool2, Integer i) {}
```



Pas nécessairement `value type`

```
record Data(Boolean bool1, Boolean bool2, Integer i) {}
```

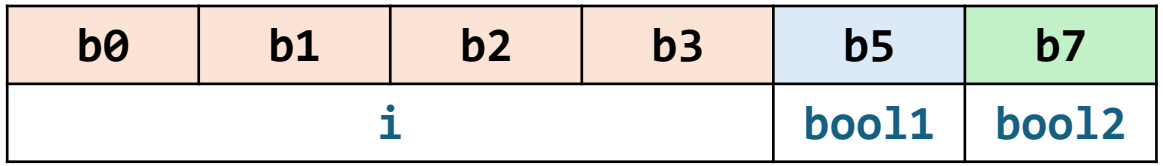


Démo

06. Null-Restricted / Layout

Aplatissement - Champs

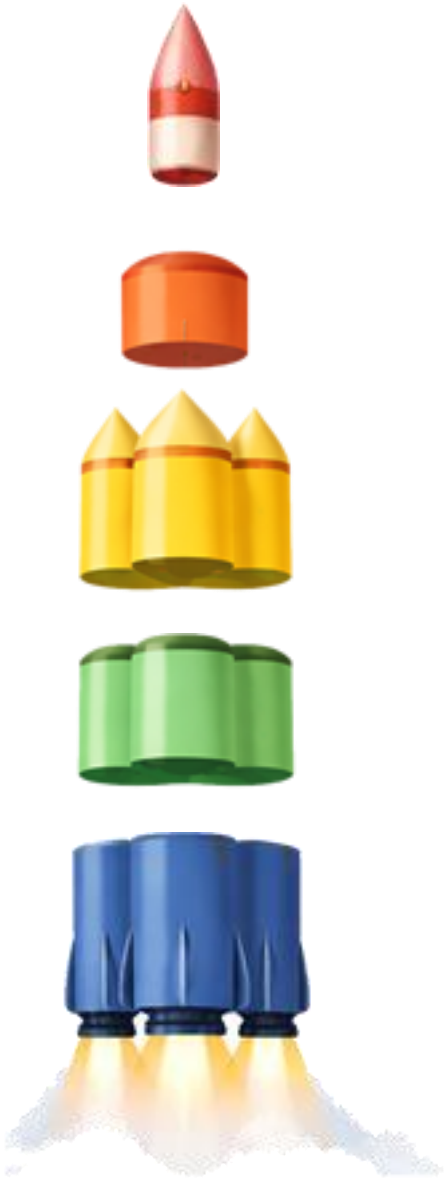
```
record Data(Boolean! bool1, Boolean! bool2, Integer! i) {}
```



Démo

07. Flattening

Roadmap Valhalla



- JEP 401 : Value Classes & Objects

- Null-Restricted and Nullable Types
- Null-Restricted Value Class Types

- JEP 402 : Enhanced Primitive Boxing (`int ≈ Integer!`)

- Parametric JVM (`List<ValueType>`)

Conclusion

Value Types

- Encapsulation (POO) + performances des primitifs
- Pas juste une “amélioration” de performance
- Combler le gap entre types primitifs et objets

https://github.com/CodeSimcoe/mandelbrot-valhalla

8317277: Java language implementation of value classes and objects #31120

View status | Code

Open MrSimms wants to merge 2819 commits into `openjdk:master` from `MrSimms:8317277`

Conversation 96 | Commits 2819 | Checks 21 | Files changed 1838 | +201220 -12533

MrSimms commented on May 11 • edited by openjdk Bot — Member

This pull request implements the first preview of JEP 401: Value Classes and Objects:

- JDK-8317277: Java language implementation of value classes and objects
  - **8317277: Java language implementation of value classes and objects** #31121
- JDK-8317278: JVM implementation of value classes and objects
  - **8317278: JVM implementation of value classes and objects** #31122
- JDK-8317279: Standard library implementation of value classes and objects
  - **8317279: Standard library implementation of value classes and objects** #31123

This pull request also includes the implementation of Strict Field Initialization in the JVM (Preview) (yet to have been assigned a JEP number). That work was implemented in the same code base because JEP 401 depends on strict field initialization.

This is the "*master pull request*" for the initial preview of JEP 401. Comments and review for a change this large will not scale well in a single pull request. This pull request serves as the vehicle for sign-off and integration into `jdk/master`. **Review comments should be directed to the appropriate "*sub-review pull request*" listed above.**

> Note
>
> The "*sub-review pull requests*" contain the same full set of code changes as this "*master pull request*" to preserve the full implementation context; the language compiler, JVM, and standard library changes are intertwined. The separate pull requests exist only to subdivide the review and related discussion by area.

Any resulting code changes should be made in `valhalla/lworld`.

`valhalla/lworld` is currently updated from `jdk/master` whenever a weekly `jdk` tag is created. At that time, code changes

Reviewers

6 more reviewers

- liach
- borodulinartm
- mgronlun
- mcimadamore
- dfuch
- viktorklang-ora

Still in progress? Learn about draft PRs

Assignees

No one assigned

Labels

build | compiler | core-libs | csr | hotspot | hotspot-jfr | i18n | javadoc | merge-conflict | net | nio | rfr | security | serviceability | shenandoah

Milestone

No milestone

Riviera DEV 2026

Les Value Types ne sont pas Complexes #Languages

Clément de Tastes

Drôle/original 😀	Très enrichissant 🤯	Super intéressant 👍	Très bon orateur 👏
Pas clair 🧐	Trop technique 🤖	Pas assez de démo/ exemple 🤔	Trop complexe 🤯

Commentaire

Votre réponse

Fig. 2. The set of C's such that $f(z) = z^2 + C$ has a stable periodic orbit.

Merci de votre attention

Démo

09. Flattening / Non-atomic