Principios de GC para JVM y Android

Víctor Orozco - @tuxtor

23 de marzo de 2019

GuateJUG

¿Porque Java?

¿Porque Java?

JAVA IS DEAD

NABENIK


Java

https://redmonk.com/sogrady/ca

Search

Besides the above plot, which can be difficult to parse even at full size, we offer the following numerical rankings. As will be observed, this run produced several ties which are reflected below (they are listed out here alphabetically rather than consolidated as ties because the latter approach led to misunderstandings). Note that this is actually a list of the Top 21 languages, not Top 20, because of said ties.

1 JavaScript
2 Java
3 PHP
4 Python
5 C#
5 C++
5 Ruby
8 CSS
9 C
10 Objective-C

NABENIK

We use cookies to analyse our traffic and to show ads. By using our website, you agree to our use of cookies.

1	1		Java
2	2		C
3	3		C++
4	5	▲	C#
5	8	▲	Python
6	7	▲	PHP
7	6	▼	JavaScript
8	12	⇈	Perl
9	18	⇈	Ruby
10	10		Visual Basic .NET

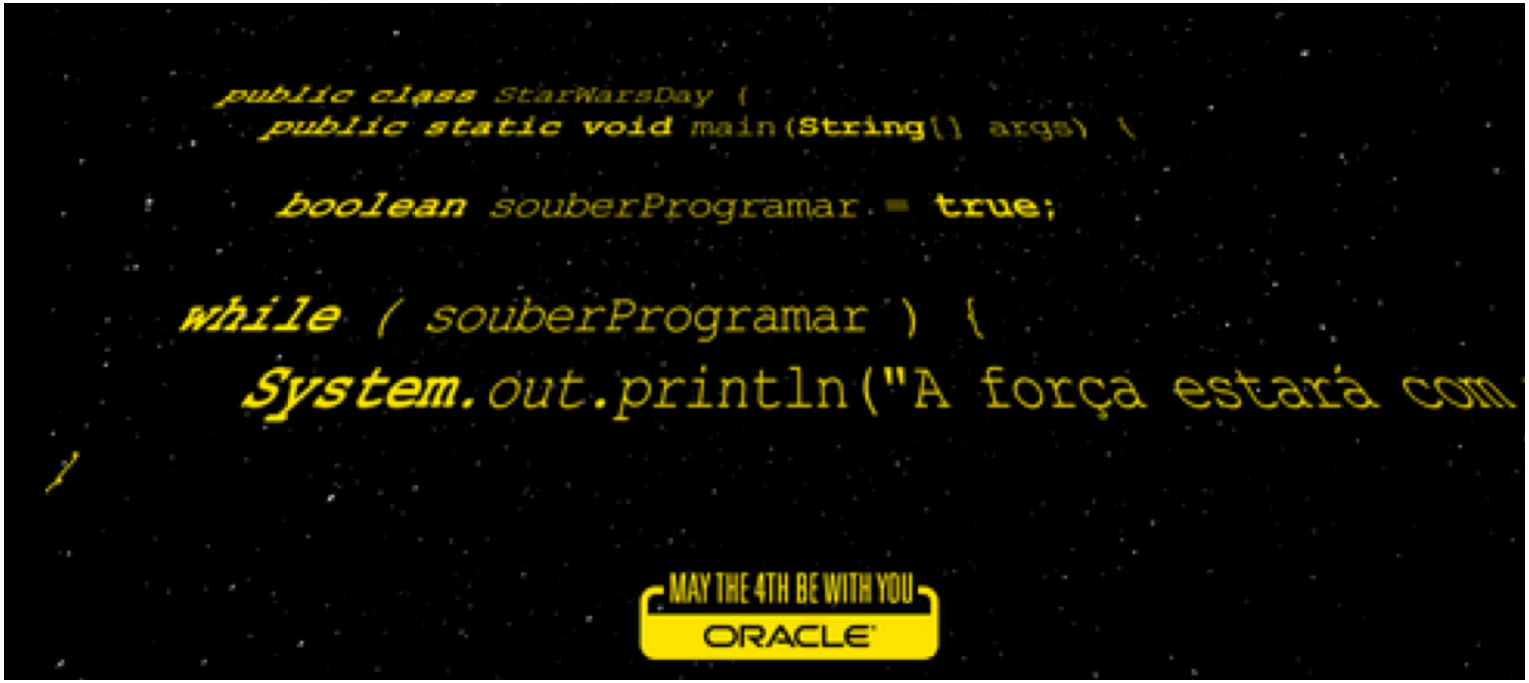
NABENIK

Java

¿Que es Java?

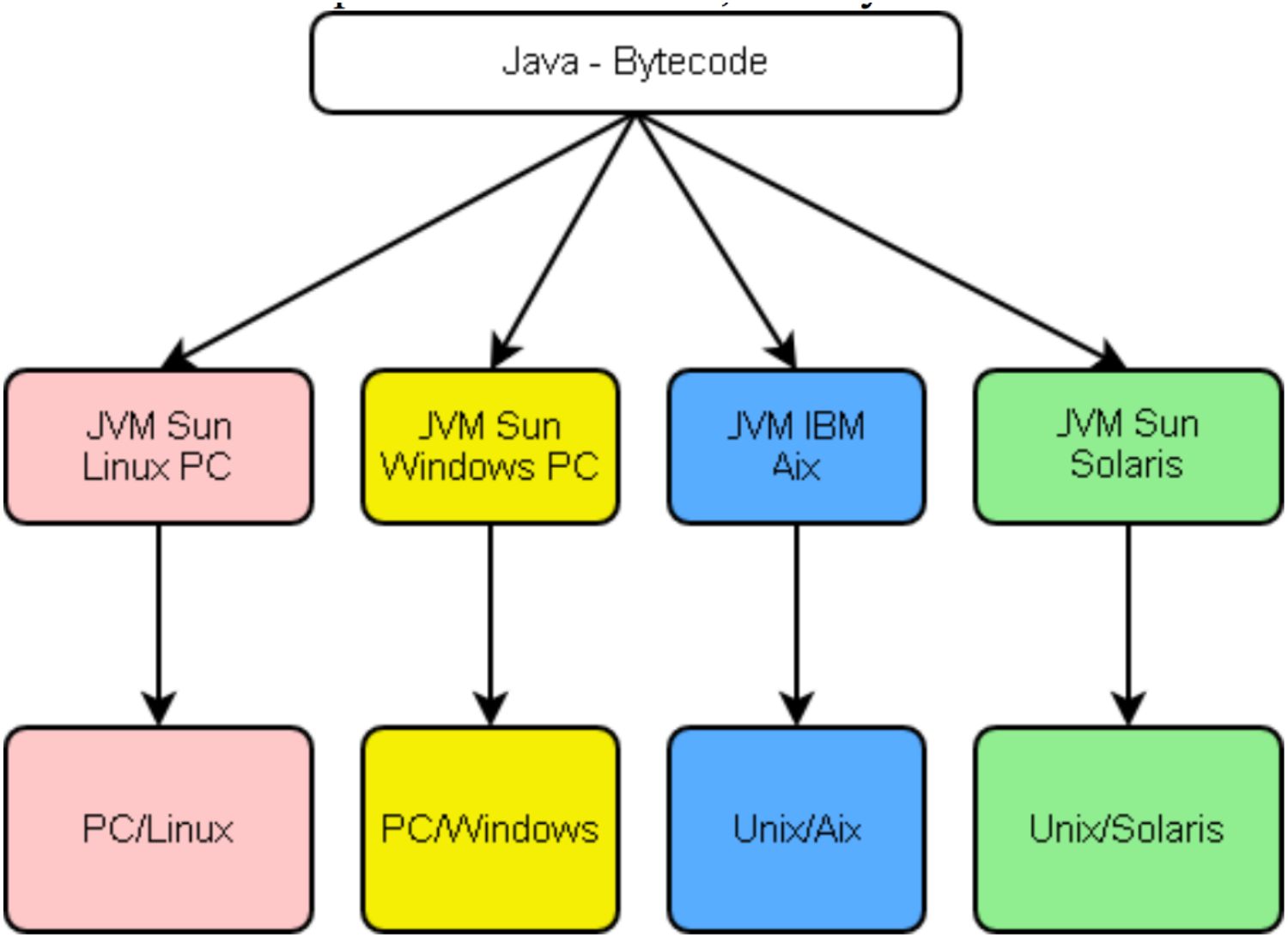
NABENIK

Lenguaje



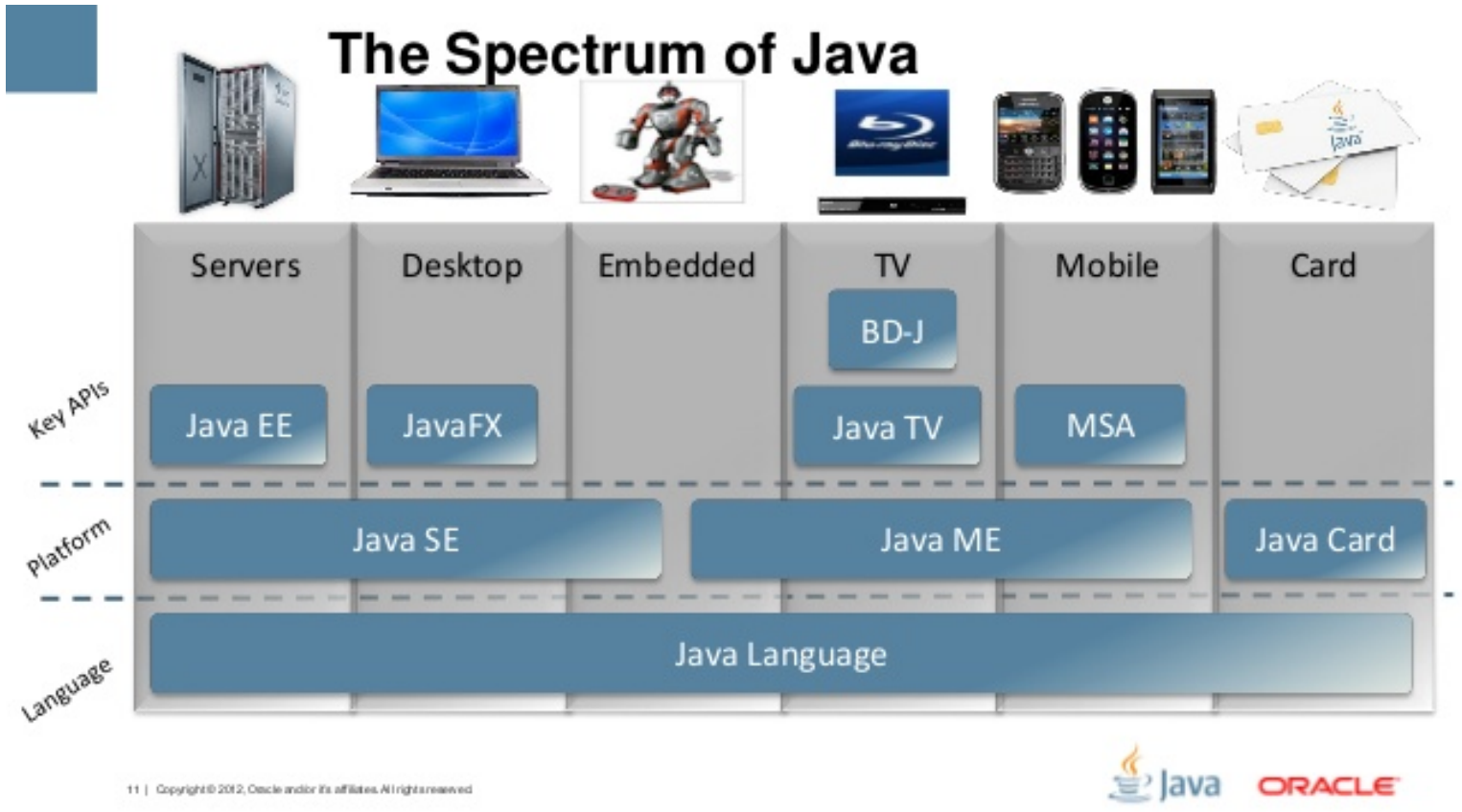
NABENIK

Maquina virtual



NABENIK

Muchas plataformas



NABENIK

¿CMS? ¿Stack heap? ¿GC Generacional? ¿Porqué mis aplicaciones explotan y Facebook no?

TLDR

Explosión

- Mala selección de tipos de dato
- Muchas variables y referencias (memory leak)
- Algoritmos innecesariamente complejos
- Muchas apps y poca memoria

Memoria en VMs

Stack vs heap

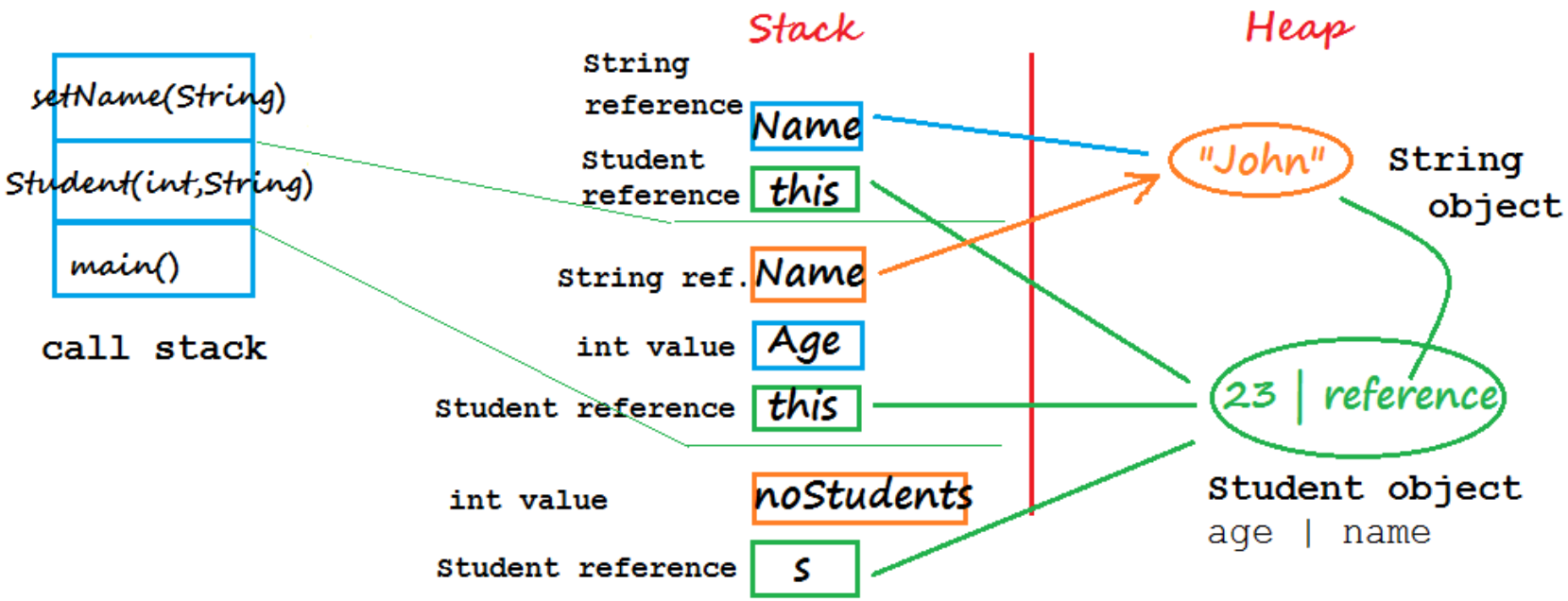


Figura 1: Stack y Heap

Stack vs heap

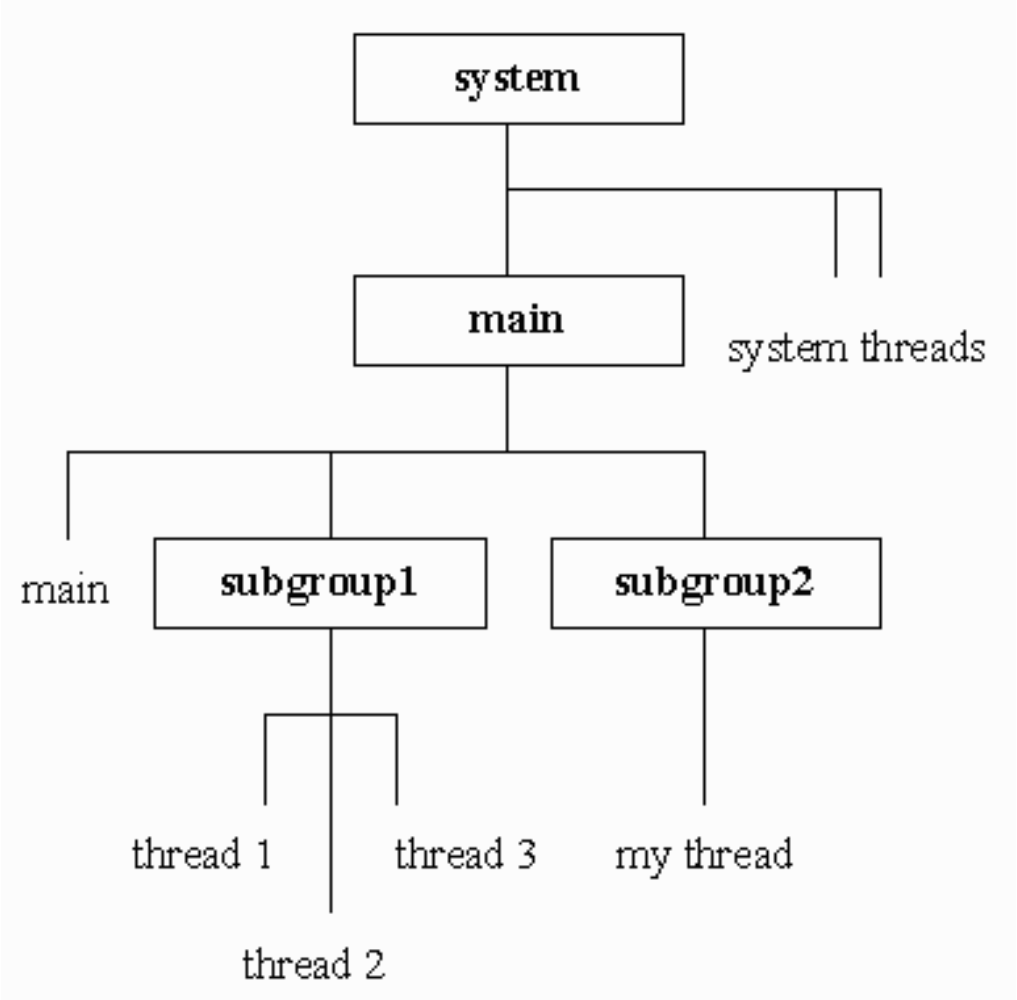


Figura 2: Java main thread

Mark and sweep

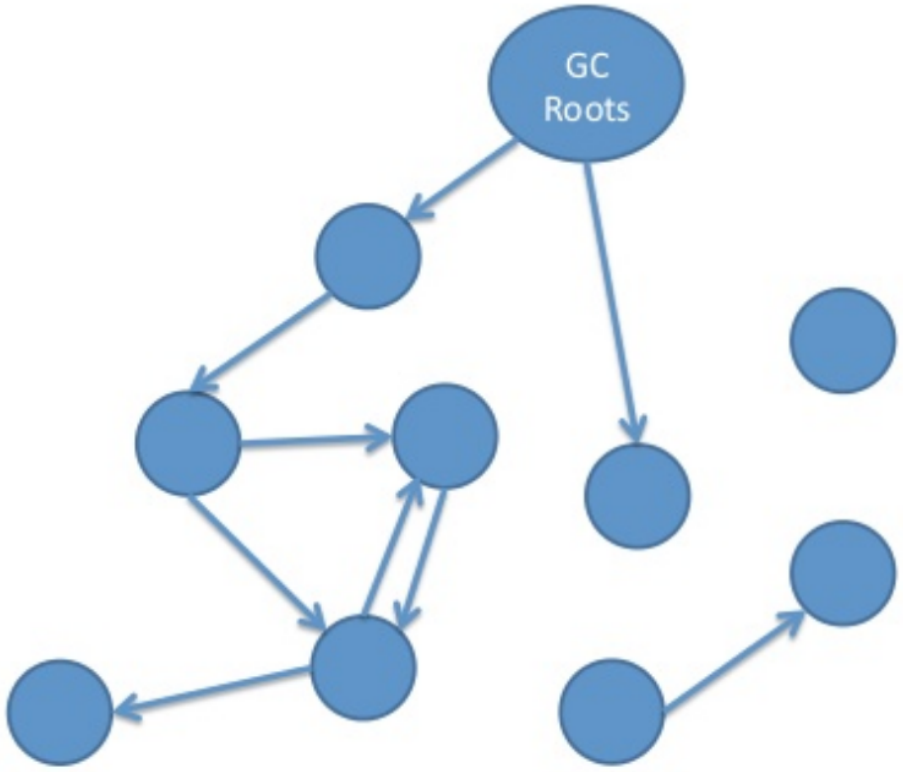


Figura 3: Mark and sweep, Credits: https://plumbr.io

Mark and sweep

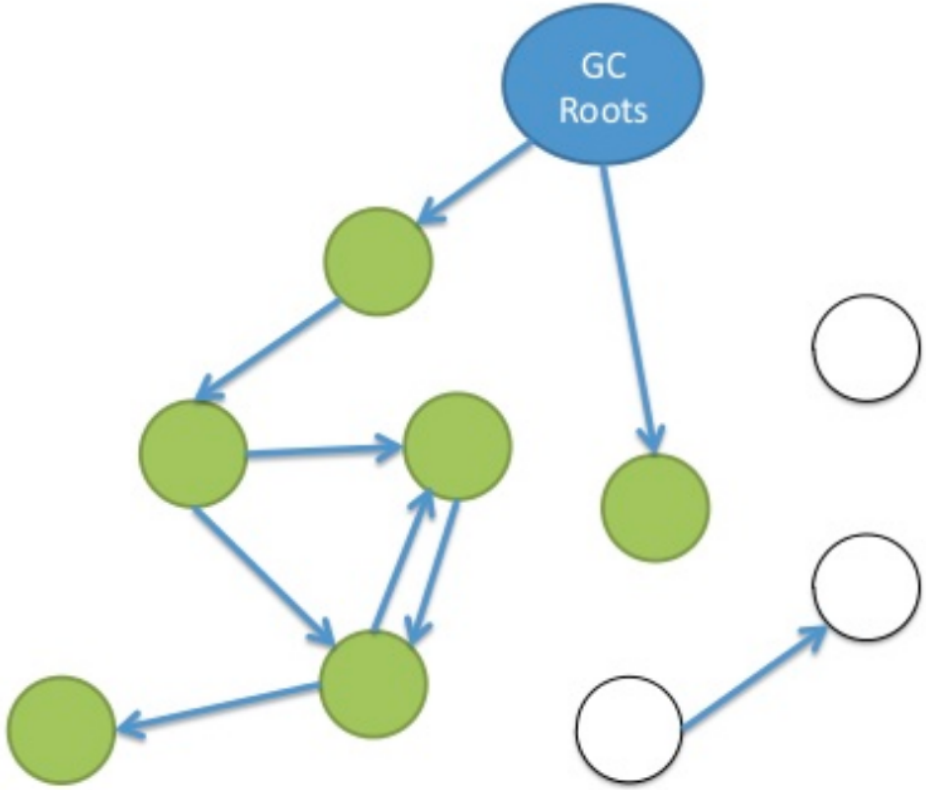


Figura 4: Mark and sweep - Mark, Credits: https://plumbr.io

Generalmente DFS

Mark and sweep

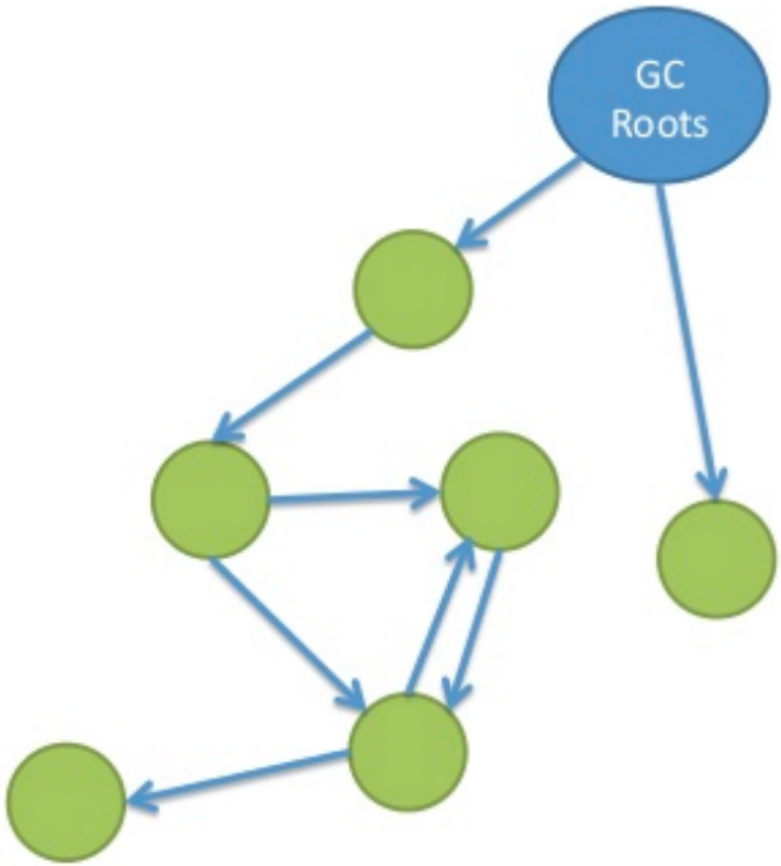


Figura 5: Mark and sweep - sweep, Credits: https://plumbr.io

Mark and sweep

Figura 6: Mark - sweep, Credits: https://plumbr.io

Mark - sweep - compact

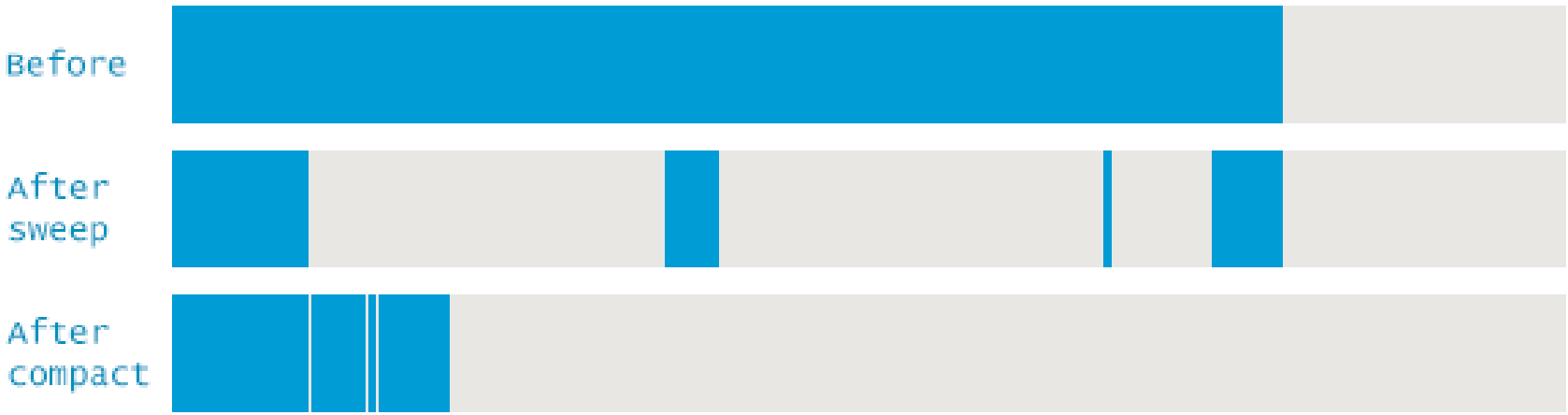


Figura 7: Mark - sweep - compact, Credits: https://plumbr.io

Mark and copy

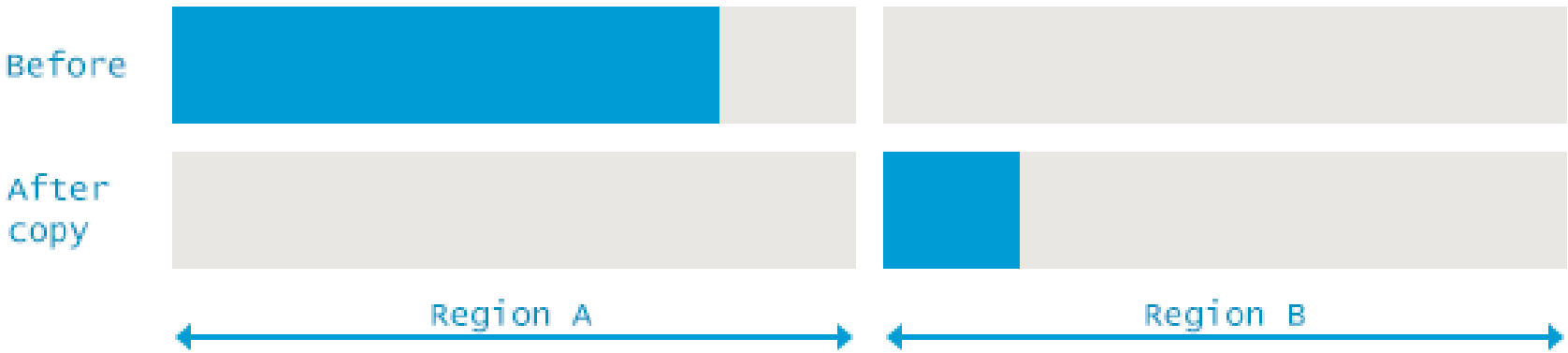


Figura 8: Mark and copy, Credits: https://plumbr.io

Demo 0 - Generación de objetos

```
//...
Stream<Integer> nums = Stream.iterate(1, n -> n + 1);

nums.forEach(num -> {
                    System.out.println(num);
                    try{Thread.sleep(10);}
                    catch(InterruptedException ex){}
          }
);
//...
```

Generational garbage collectors

Generational GC(HotSpot)

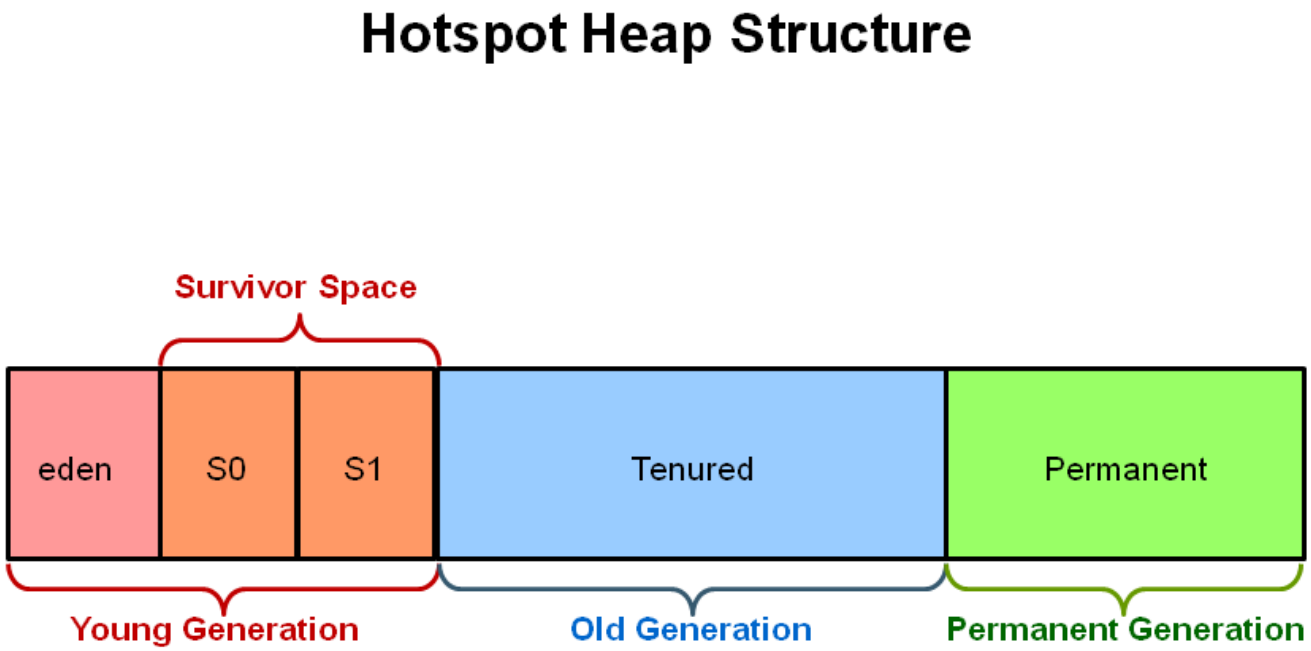


Figura 9: Generational GC, Credits: Oracle

Generational GC(HotSpot)

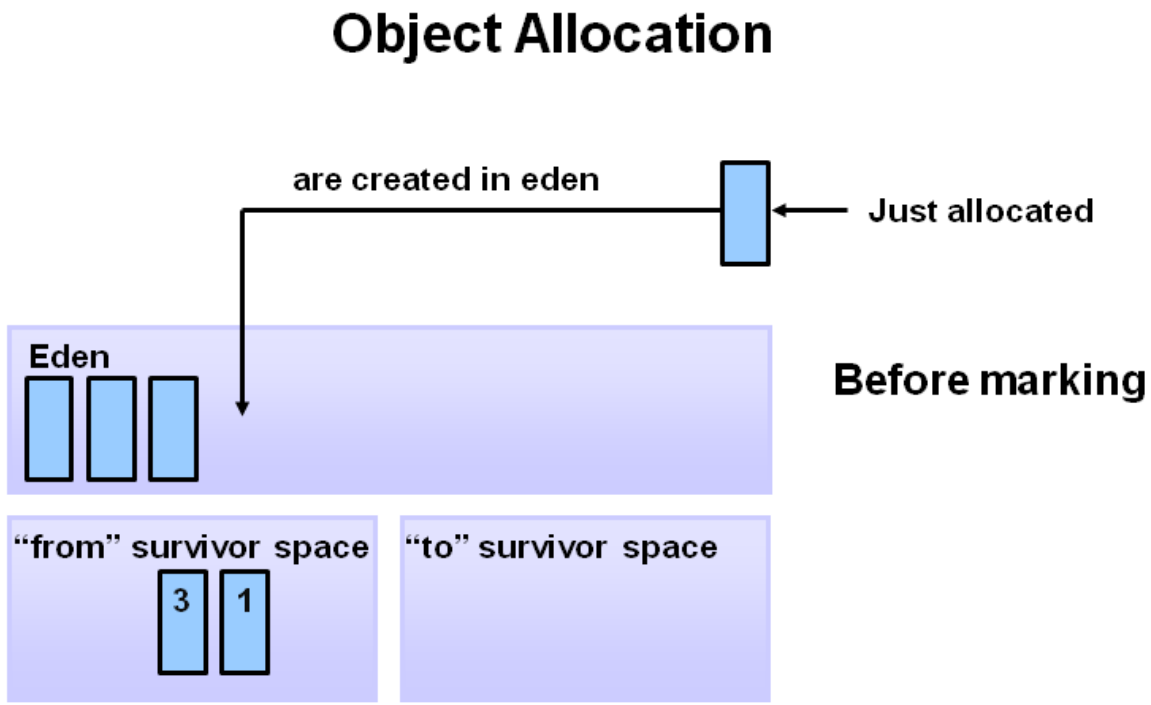


Figura 10: Generational GC, Credits: Oracle

Generational GC(HotSpot)

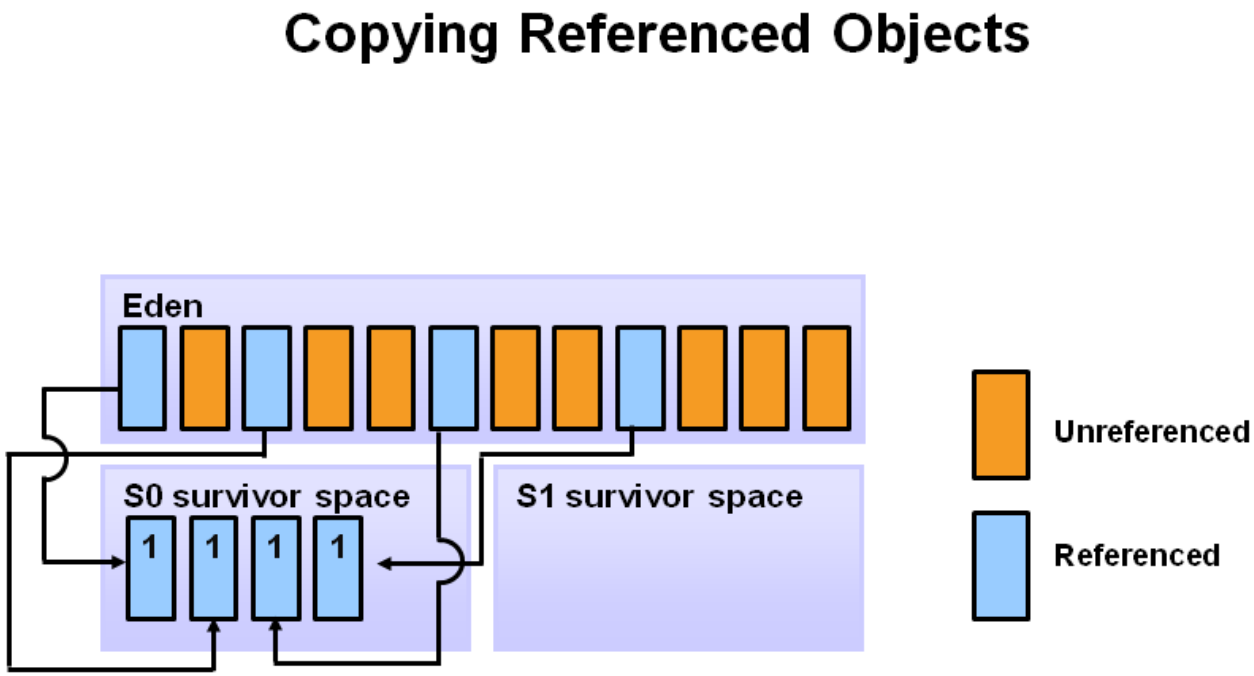


Figura 11: Generational GC, Credits: Oracle

Generational GC(HotSpot)

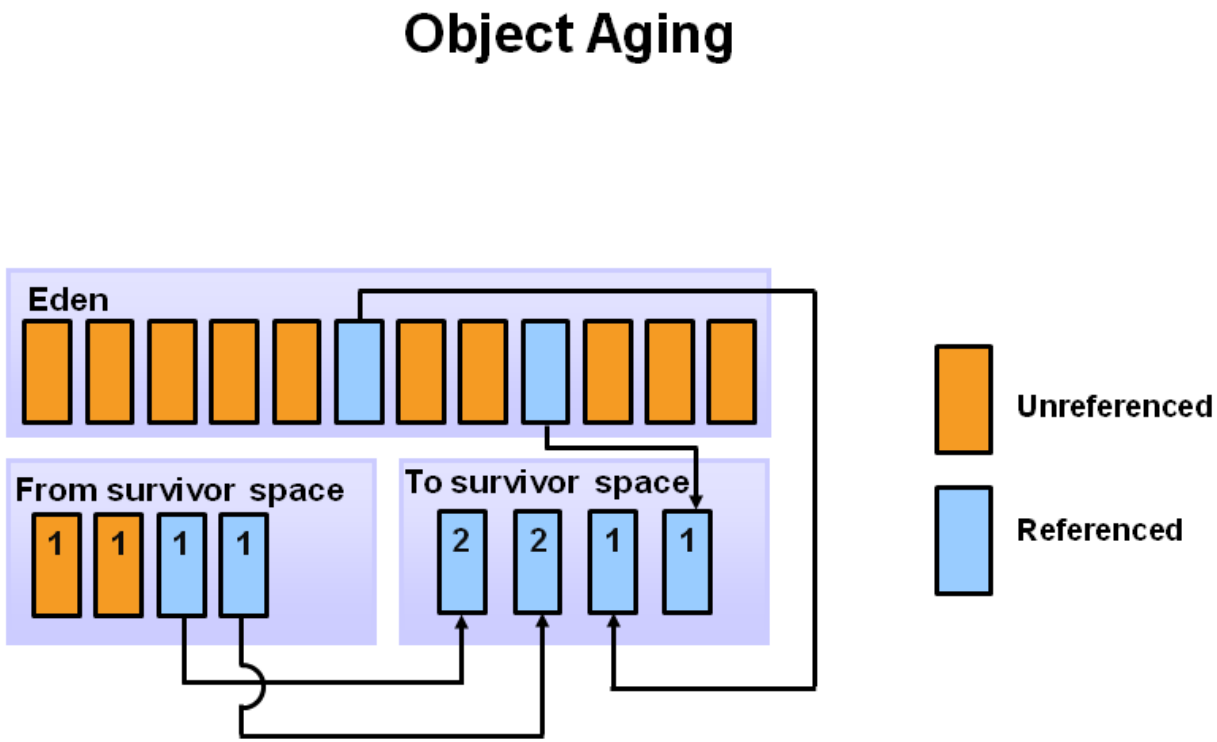


Figura 12: Generational GC, Credits: Oracle

Generational GC(HotSpot)

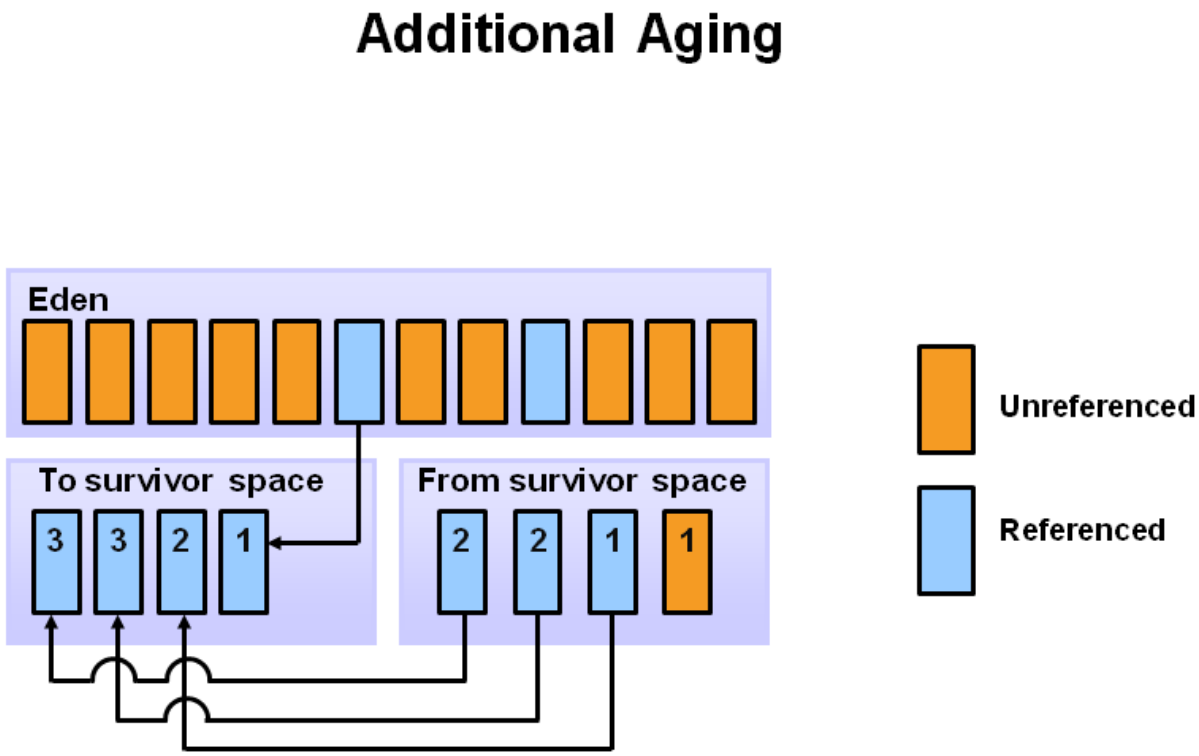


Figura 13: Generational GC, Credits: Oracle

Generational GC(HotSpot)

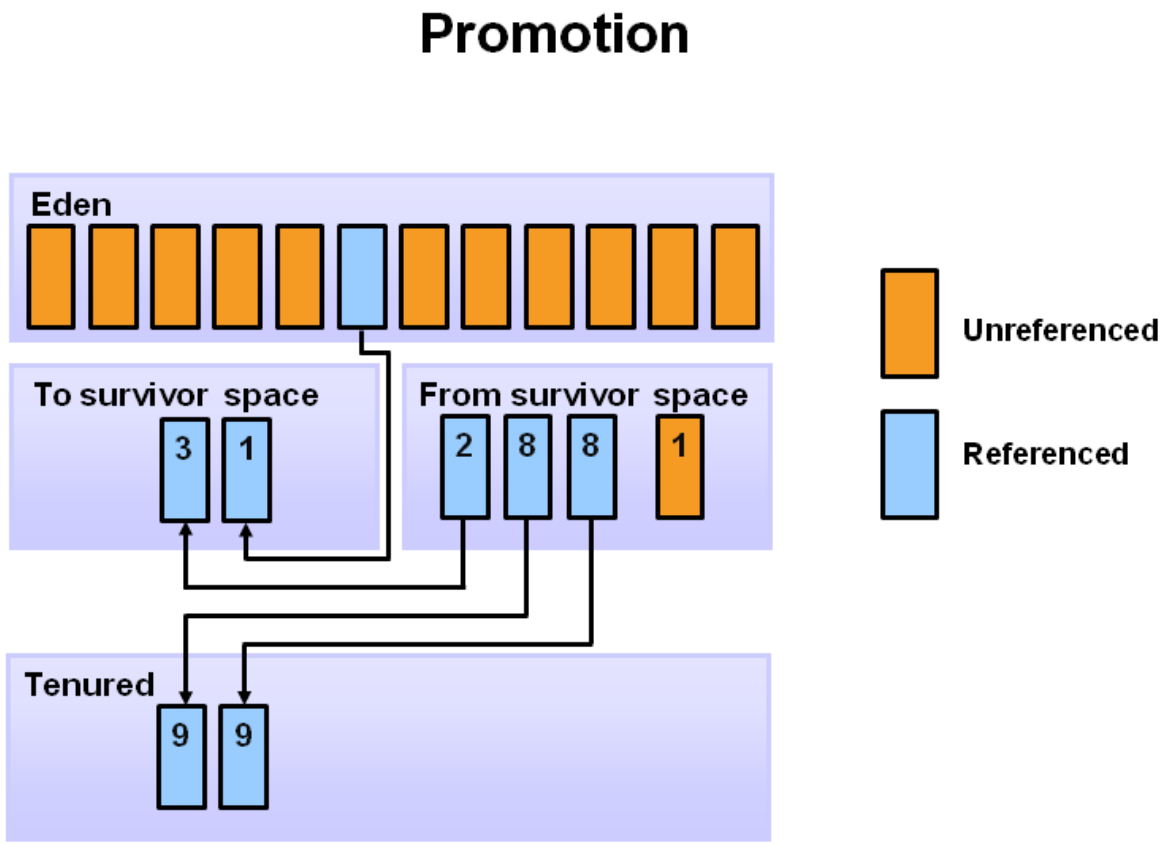


Figura 14: Generational GC, Credits: Oracle

Generational GC(HotSpot)

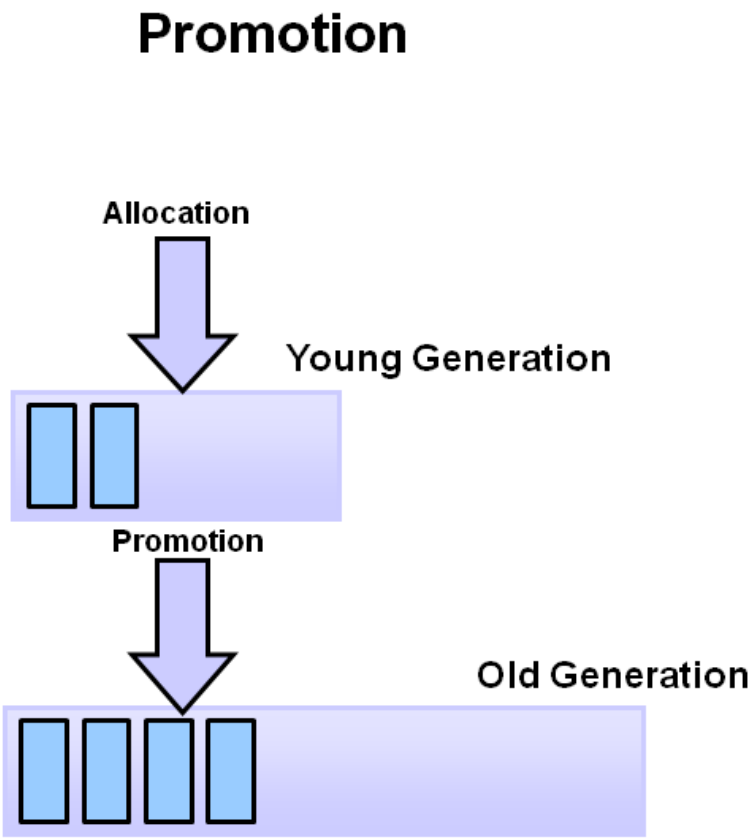


Figura 15: Generational GC, Credits: Oracle

Demo 1 - GC Generacional

```
//...
Stream<Integer> nums = Stream.iterate(1, n -> n + 1);

var numeros = new ArrayList<Integer>();

nums.forEach(num -> {
        System.out.println(num);
        numeros.add(num);
});
//...
```

java -XX:+UseSerialGC GenerationsDemo

GC en OpenJDK y Android

Recolectores de basura

OpenJDK

- Serial GC para Young y Old generations
- Parallel GC para Young y Old generations
- Parallel New para Young + Concurrent Mark and Sweep (CMS) para Old Generation
- G1GC, para Young y Old generations

Android

- Sticky CMS
- Partial CMS
- Full CMS
- RosAlloc (Slots Allocator)

Tip: Ustedes no controlan la ejecución del GC, solo los OEM

SerialGC

- Young: Mark-Copy
- Old: Mark-Sweep-Compact
- $java - XX : +UseSerialGCcom.nabenik.MyExecClass$

ParallelGC

- Young: Mark-Copy
- Old: Mark-Sweep-Compact
- Stop-the-world en ambas regiones
- $java - XX : +UseParallelGCcom.nabenik.MyExecClass$

NABENIK

CMS

- Young: Mark-Copy - Stop the world
- Old: (Mostly)Concurrent Mark Sweep (Paralelo)
- $java - XX$:
  $+UseConcMarkSweepGCcom.nabenik.MyExecClass$

ART - CMS

- Sticky CMS (non-moving)
- Compact = Imperceptible (background/cached)
- Young es reemplazado por Allocation Stack (java.lang.Object)
- Activity Manager

Explosión

- Broadcast Receiver 10 segundos
- Activity Manager 5 segundos

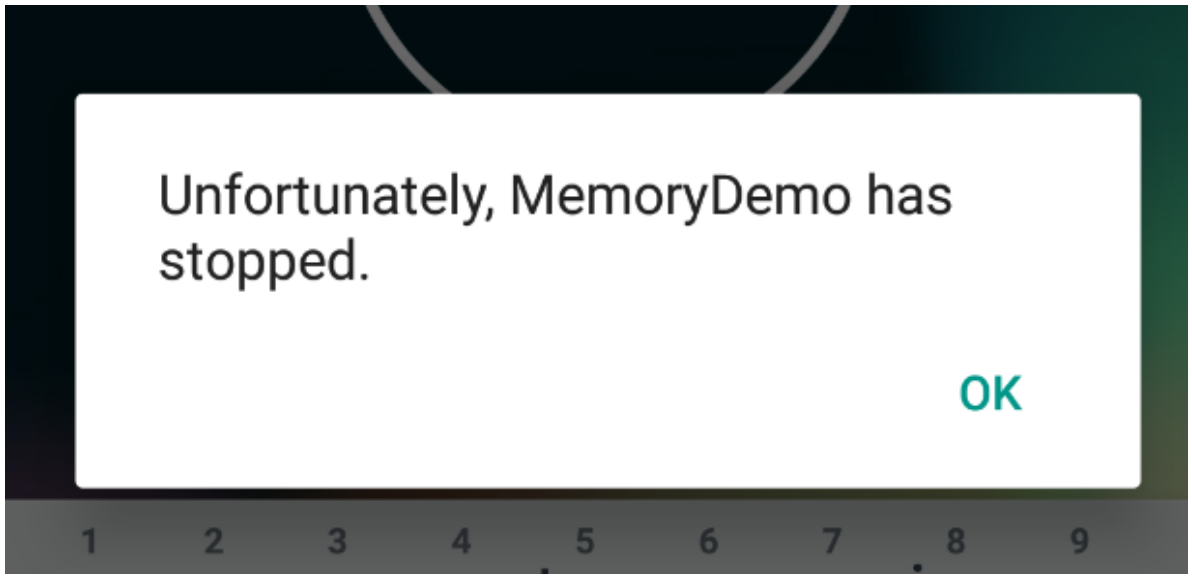


Figura 16: Dialogo ANR

ART - CMS

```
<application
...
android:largeHeap="true"
...
</application>
```


NABENIK

Como luchar CONTRA un Garbage Collector

Demo 1 - Referencias + Mal tipo de dato

```
//...
var lasReferencias = new ArrayList<String>();
Stream<Integer> numeros = Stream.iterate(1, n -> ++n);

numeros.forEach(n -> {
                    lasReferencias.add(n + "");
                    if(lasReferencias.size() % 10_000_000 == 0)
                    System.out.println(n);
                    try{ Thread.sleep(3000); }
                    catch(InterruptedException e){}
                    }
          });
}
//...
```

Demo 2 - Referencias

```
//...
var lasReferencias = new ArrayList<Integer>();
var numeros = IntStream.iterate(1, n -> ++n);

numeros.forEach(n -> {
                    lasReferencias.add(n);
                    if(lasReferencias.size() % 10_000_000 == 0)
                    System.out.println(n);
                    try{ Thread.sleep(3000); }
                    catch(InterruptedException e){}
          }
});
//...
```


NABENIK

Demo 3 - Concatenación de String

```
//...
static String texto = "";
public static void main(String[] args){
        var numeros = IntStream.iterate(1, t -> ++t);

        numeros.forEach(n -> {
                texto += "␣" + n;
                if(n % 10_000 == 0) System.out.println(n);
        });
}
//...
```

Demo 4 - StringBuffer

```
//...
static StringBuilder texto = new StringBuilder("");
public static void main(String[] args){
        var numeros = IntStream.iterate(1, t -> ++t);

        numeros.forEach(n -> {
                texto.append("␣");
                texto.append(n);
                if(n % 10_000 == 0) System.out.println(n);
        });
}

//...
```

Complejidad de algoritmos

Complejidad

Complejidad = Cantidad de pasos para realizar una tarea

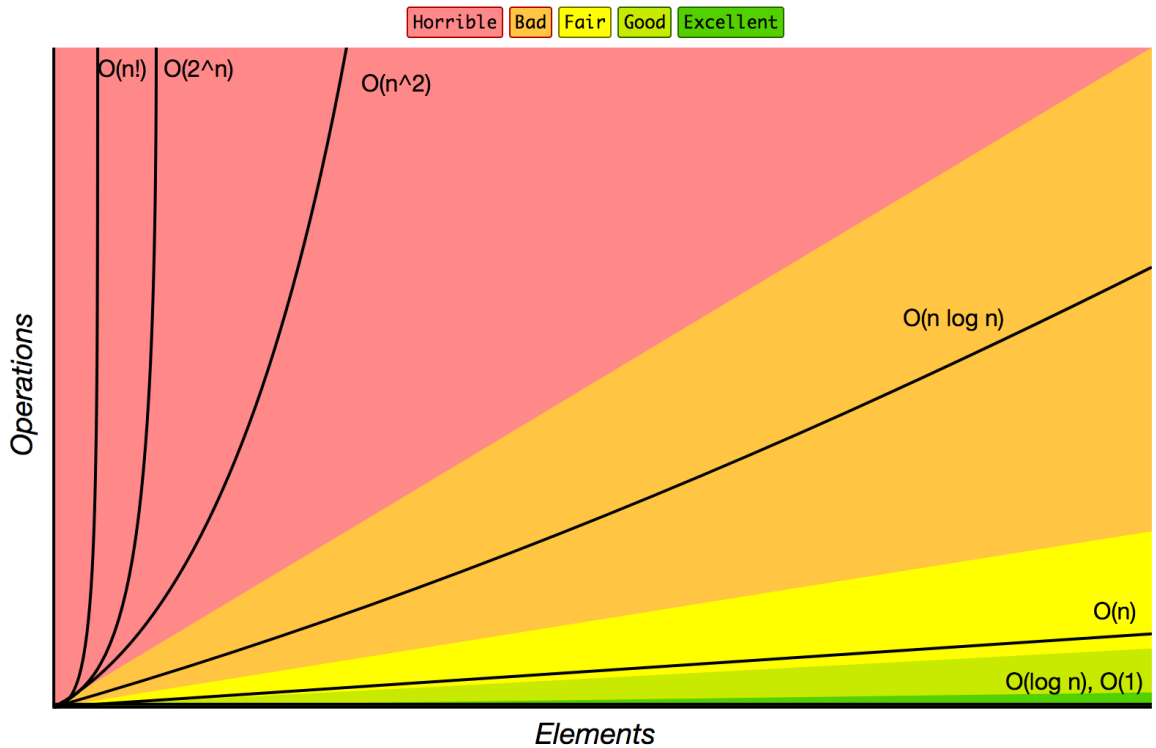


Figura 17: Complejidad computacional

También conocido como "programar bien" :O

Fibonacci

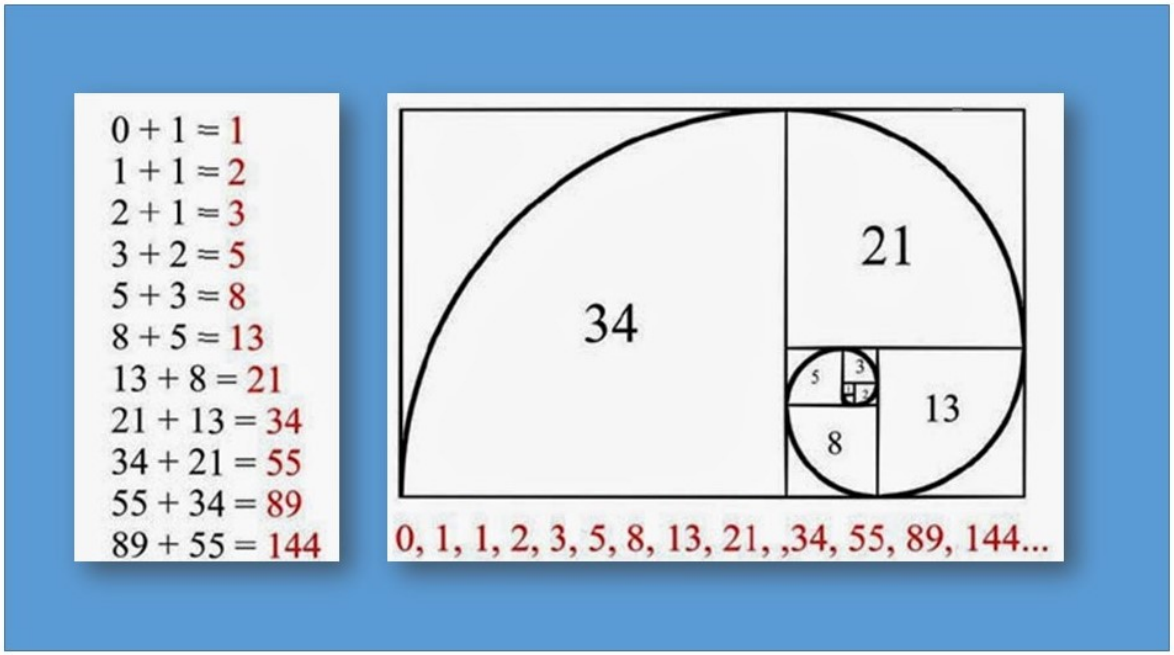


Figura 18: Sucesión Fibonacci

Demo 5 - Mal Fibonacci

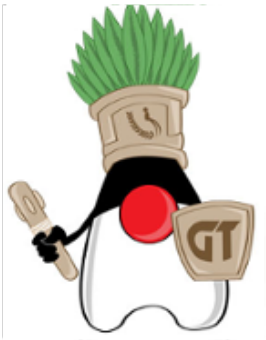
```
//...
public static long doFibonacci(int n) {
        if (n <= 1)
                return n;
        else
                return doFibonacci(n-1) + doFibonacci(n-2);
}
//...
```

Demo 6 - Buen Fibonacci

```
//...
public static long doFibonacci(int n) {
        long a=0, b=1, c=0;

        for(int i = 0 ; i < n; i++){
                c = a + b;
                a = b;
                b = c;
        }
        return c;
}
//...
```

Víctor Orozco



- me@vorozco.com
- @tuxtor
- http://vorozco.com
- http://tuxtor.shekalug.org

NABENIK