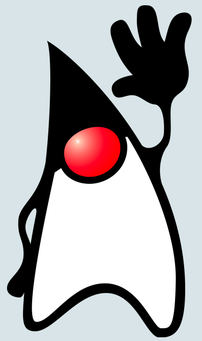
СБЕРБАНК

Корпоративный университет



Углубленные основы

Занятие №19

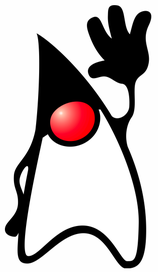
СБЕРБАНК
Корпоративный университет

Куценко Виталий

Маршрут

СБЕРБАНК
Корпоративный университет

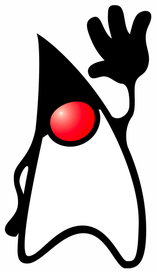
- Remote debug
- Примитивные типы
- ByteCode, ClassLoader
- Сборщики мусора



Договорённости

СБЕРБАНК
Корпоративный университет

- Активно участвуем. Не стесняйтесь задавать вопрос.
- Но off-topic обсуждаем в Telegram @sb_ku_java_2019_10
- Не стесняйтесь просто спрашивать в telegram.
-

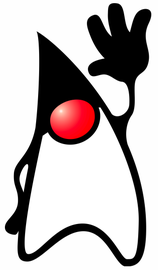


Договорились?
Поехали!

01

Remote Debug

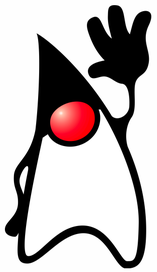
- Смотрим код, что и как



Remote Debug. Hot Swap

СБЕРБАНК
Корпоративный университет

- Стандартный HotSwap имеет много ограничений
- Более мощное средство - jRabel. Основан на механизме ClassLoader
- Есть «дешевый сердитый» аналог - «Spring Loader»
- Может отслеживать изменения .class-файлов и «на лету» обновлять их в JVM. Чревато «утечками памяти»



Ваши вопросы?

02

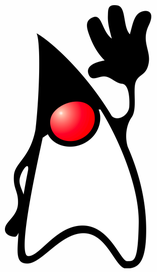
Примитивные типы в Java

Примитивные типы

СБЕРБАНК
Корпоративный университет

Тип	Размер, биты
byte	8
short	16
int	32
long	64
float	32
double	64
char	16
boolean	?

Так говорит JLS:
https://docs.oracle.com/javase/specs/jls/se11/html/jls-4.html#jls-4.2



Структура объекта

Объект состоит из:

заголовка

блока с данными.

Заголовок не описан в спецификации, но о нем есть информация.

object header

состоит из двух частей:

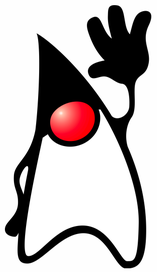
mark word - статус синхронизации, хэш объекта, возраст для gc.

klass pointer - указывает на "класс-описатель" объекта.

Для любителей копнуть глубже… (markOop.hpp The markOop describes the header of an object.)

И еще… klass.hpp A Klass provides: 1: language level class object (method dictionary etc.)

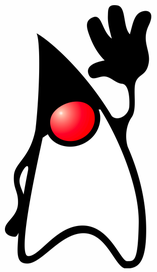
2: provide vm dispatch behavior for the object



String

СБЕРБАНК
Корпоративный университет

- String хранится в отдельной области памяти JSP
- String x = «Тест» и String x = new String(«Тест») - не одно и то же
- Java <= 6 - PermGen, Java >= 7 - Heap
- Java <= 8 - char[], Java >=9 - char[] или byte[]



Ваши вопросы?

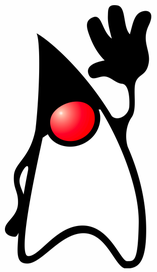
03

ByteCode, Classloader

Как работает JVM

Программа на java из исходного кода компилируется в byte-code.

Byte-code выполняется в JVM.



Как работает JVM

А оно работает так:

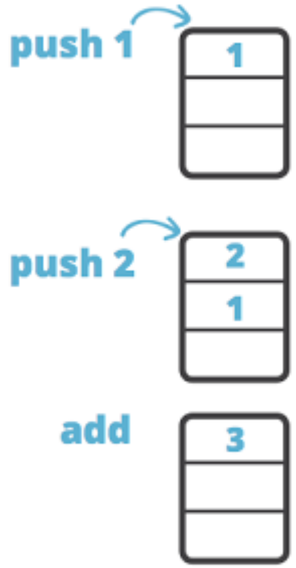
Да, JVM – это стек.

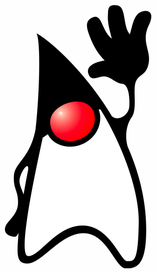
x + y это в byte-коде:

iload_1 – загрузить в стек x

iload_2 – загрузить в стек y

iadd – сложить, два значения из стека





Что такое Byte-коды

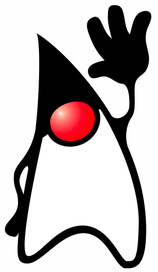
СБЕРБАНК
Корпоративный университет

Byte-коды – это инструкции для JVM, длиной в 1 байт.
Всего их может быть 256.

Примеры:
iload – загрузить в стек локальную переменную int
fstore – выгрузить из стека в локальную переменную float
ladd – сложить два long-а не стеке.

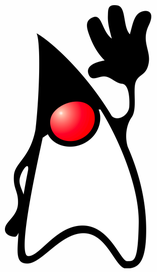
Операции для boolean, byte, char, short не определены, они приводятся к int.

Кстати, а какой размер boolean согласно спецификации JVM?



Byte-коды. Пример

Пример Byte-кодов:
class OneOne

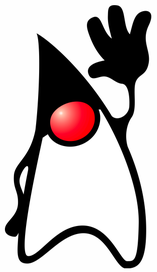


ASM

ASM is an all purpose Java bytecode manipulation and analysis framework.

Asm используется в следующих проектах:

- OpenJDK
- Nashorn compiler
- Groovy compiler
- Kotlin compiler
- Cobertura и Jacoco
- CGLIB
- Gradle



Class-loader

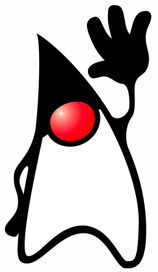
A class loader – это объект, который отвечает за загрузку классов.

Фактически это «abstract class ClassLoader».

```
ClassForLoading obj = new ClassForLoading();
```

можно представить как:

```
Class<?> clazz = ClassLoader.defineClass("ClassForLoading")
Constructor<?> constructor = clazz.getConstructor();
ClassForLoading obj = constructor.newInstance();
```



Как загружаются классы:

JVM Specification: Chapter 5. Loading, Linking, and Initializing

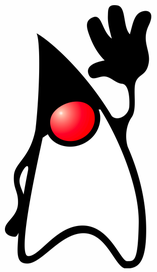
В спецификации JVM определяются два типа загрузчиков:

bootstrap class loader (входит в состав Java Virtual Machine)

user-defined class loaders (наследник class loader).

Важно.
Во время выполнения класс или интерфейс определяются парой:

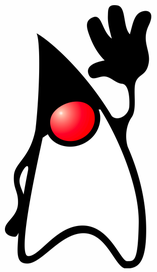
Class loader – class name.



Class Loader. Пример

Пример простого class loader-а:

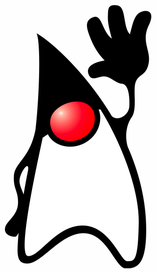
MyClassLoaderDemo



Самодельный класс

Hello, World из байт-кодов.

Пример ASMdemoCreateClass.



Ваши вопросы?

04

Сборщики мусора

OTUS

Сборщик мусора. Введение

Для чего нужен сборщик мусора?

Нужен, чтобы собирать мусор,
т.е. ненужные объекты,
т.е. объекты, на которые никто не ссылается.

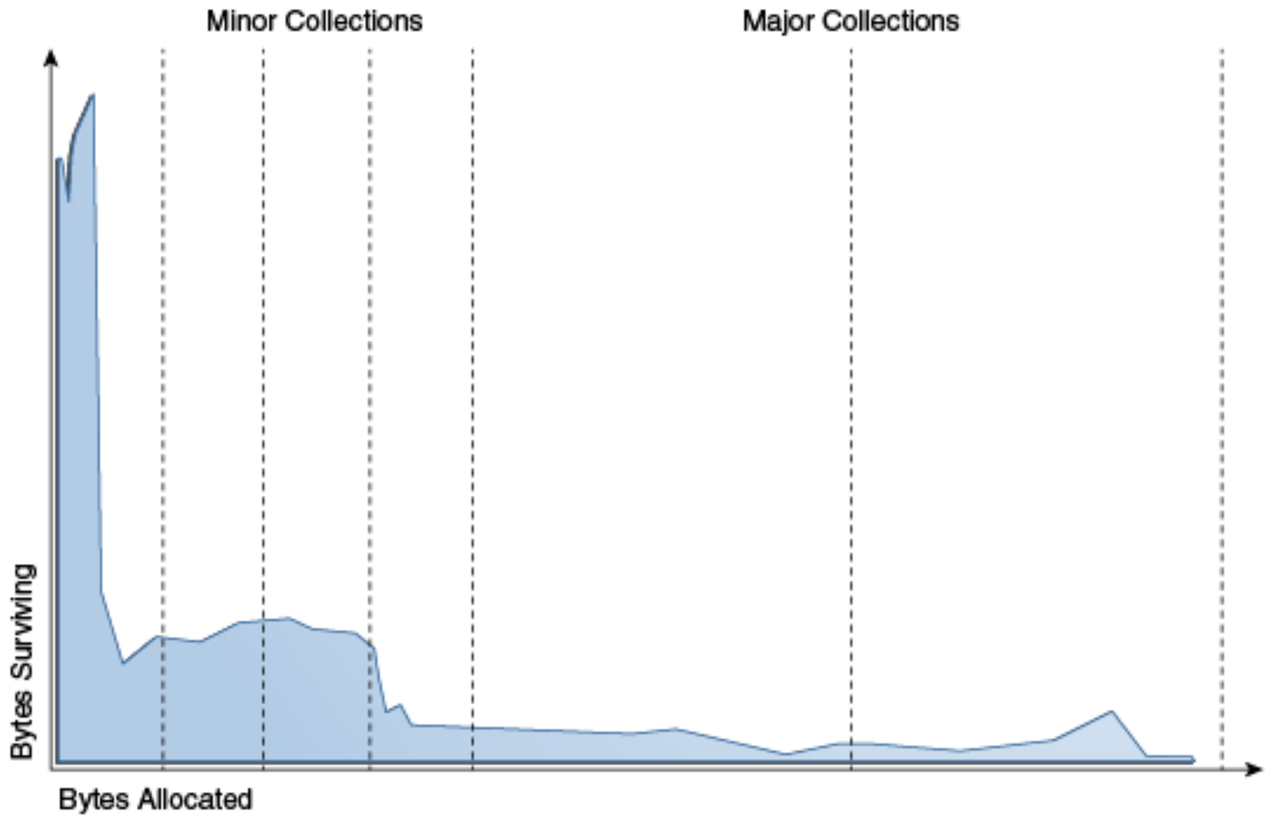
А кто может ссылаться?

Garbage Collection Roots:

- Local variables
- Active threads
- Static fields
- JNI references

и т.д.

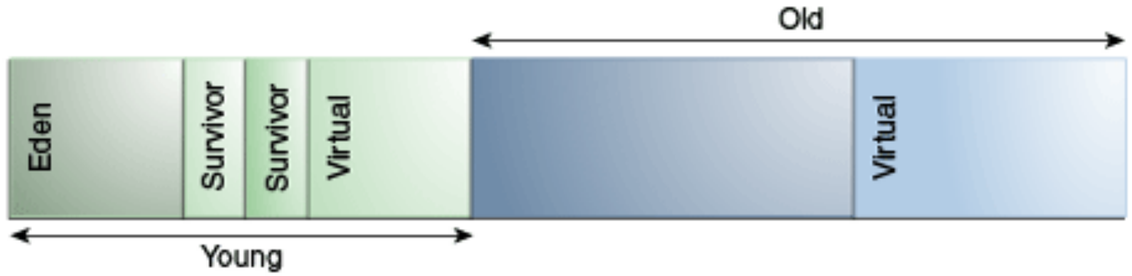
Теория поколений



Объекты умирают молодыми!

Делим объекты на поколения

Молодое поколение и старое поколение

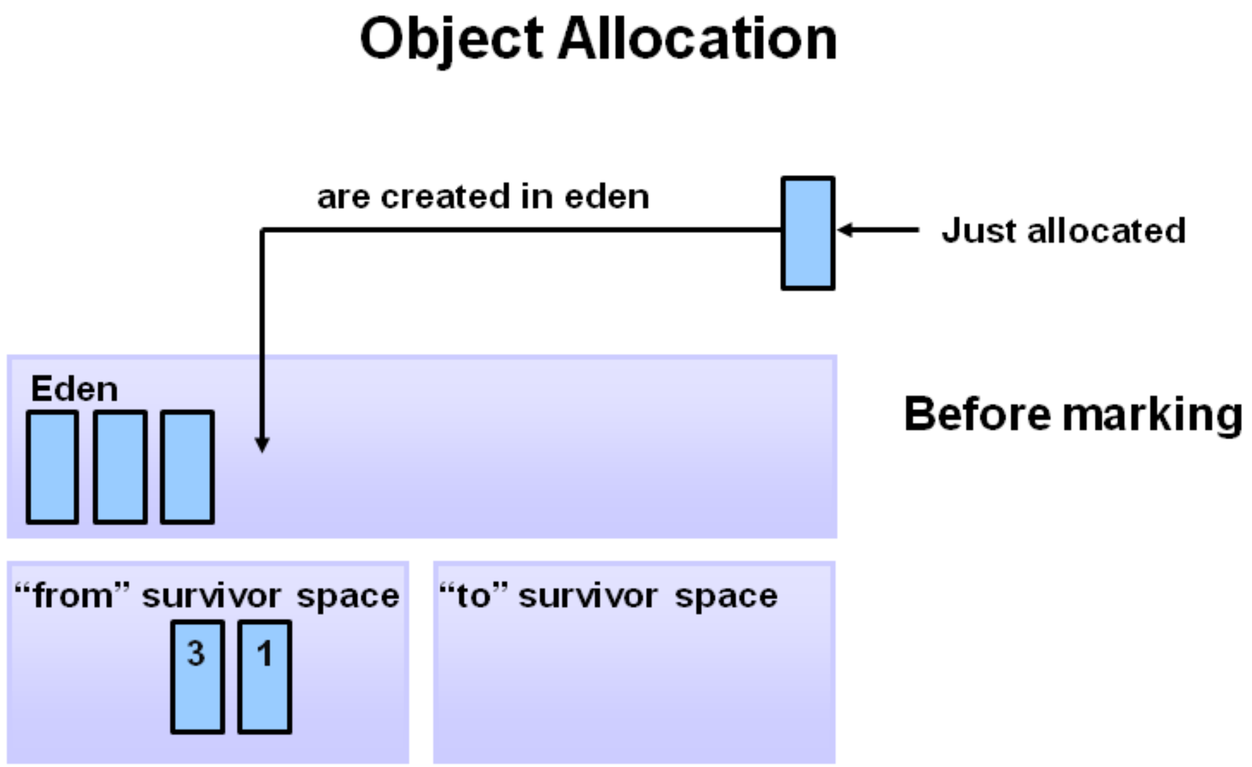


Молодое поколение делится на:

- Eden
- Survive from
- Survive to

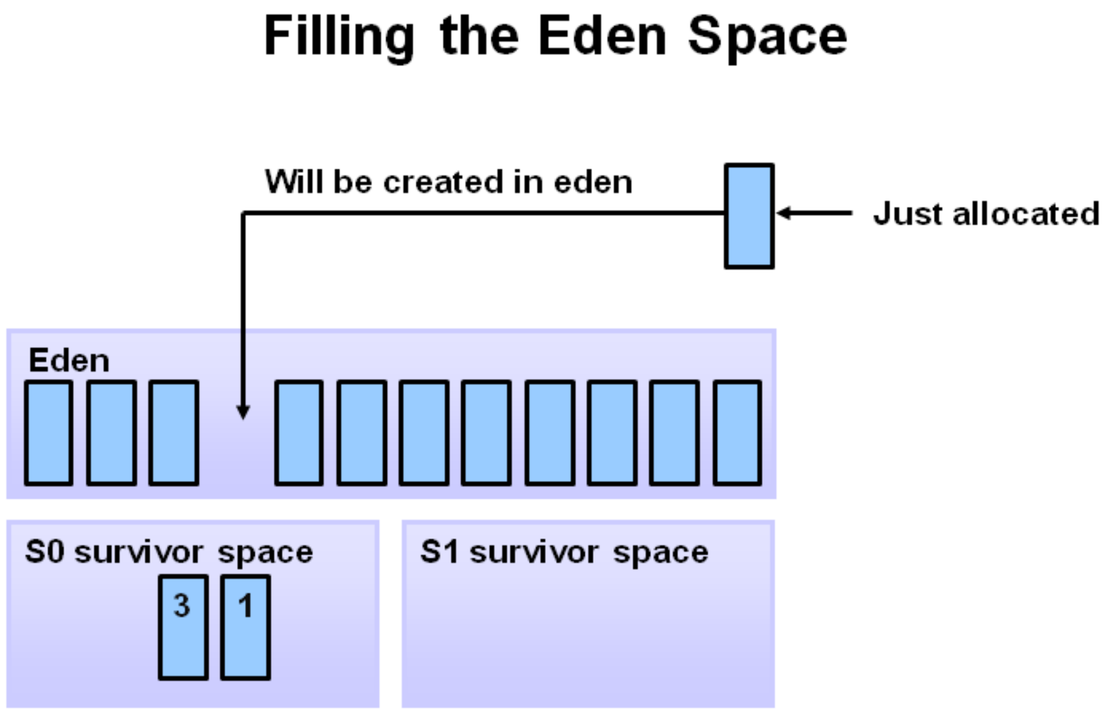
«Карьера» объекта в Java

1. Объект создается в «молодом» поколении



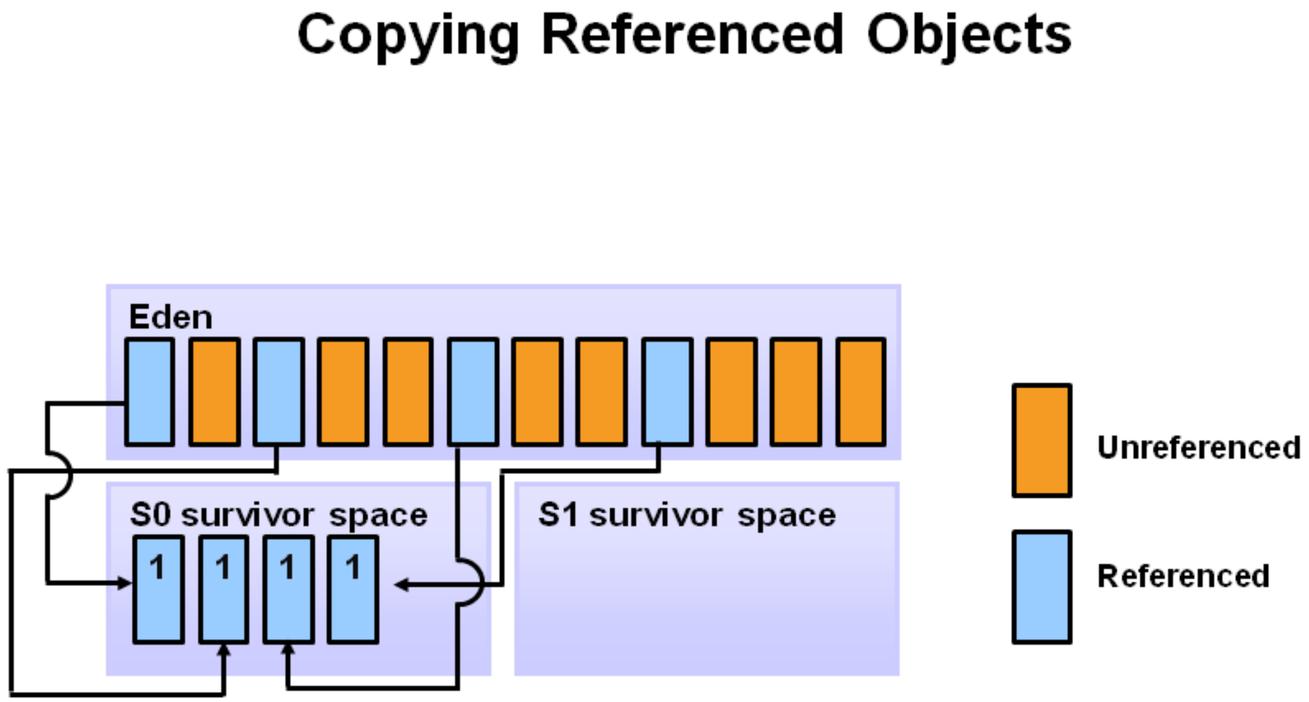
«Карьера» объекта в Java

2. Когда место в Eden заканчивается,
запускается minor-ная сборка



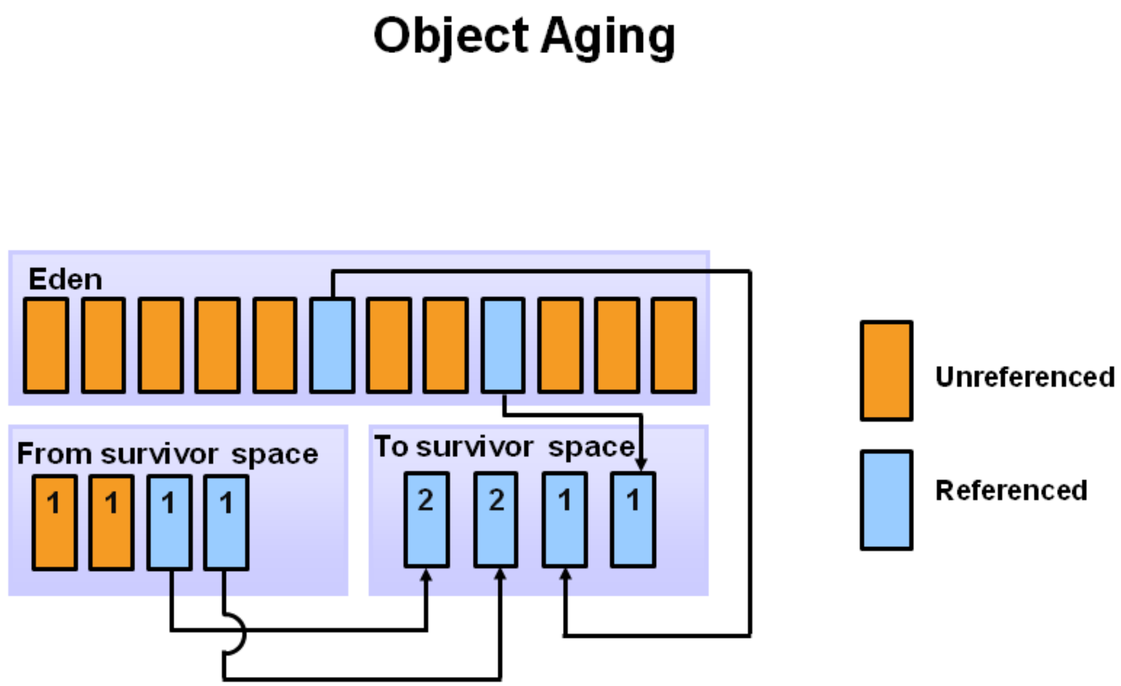
«Карьера» объекта в Java

3. Нужные (на которые есть ссылки) объекты переносятся в Survivor “to”



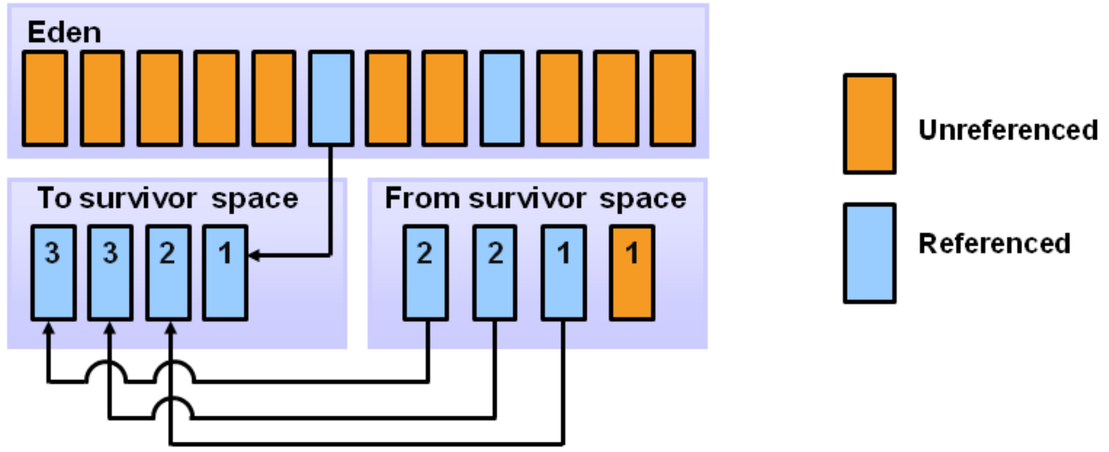
«Карьера» объекта в Java

4. При следующей minor-ной сборки объекты из Eden переносятся в Survivor “to”, и туда же переносятся объекты из Survivor “from”, но их возраст увеличивается (где хранится «возраст»?).



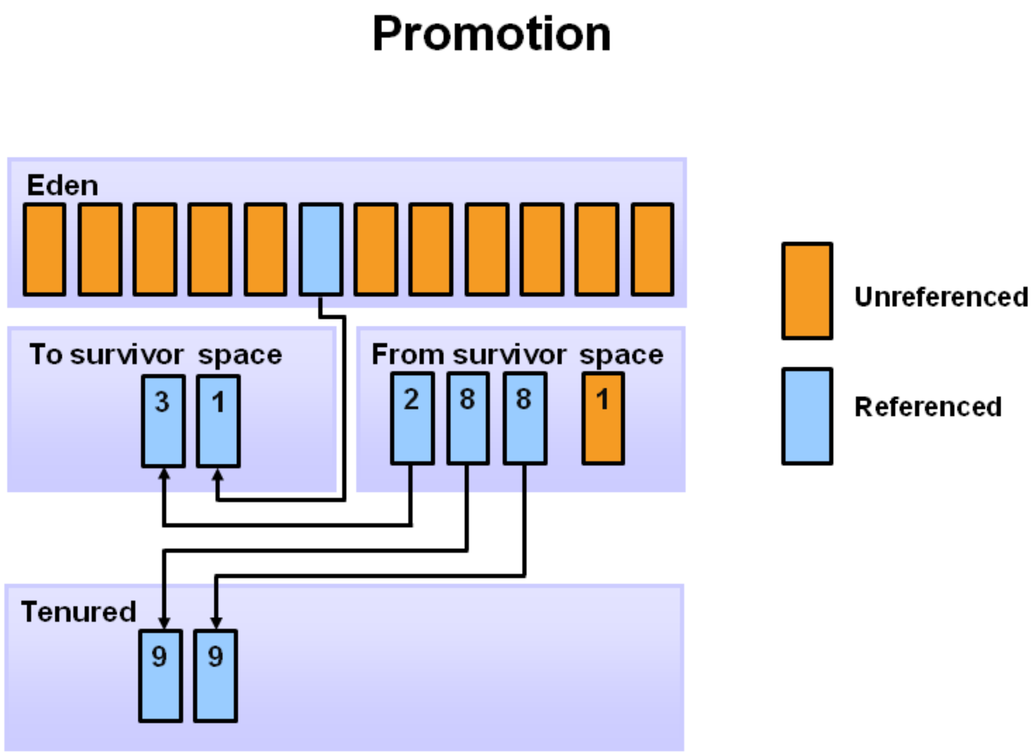
«Карьера» объекта в Java

5. При следующей minor-ной сборке все повторяется,
но Survivor “to” и Survivor “from” меняются местами.



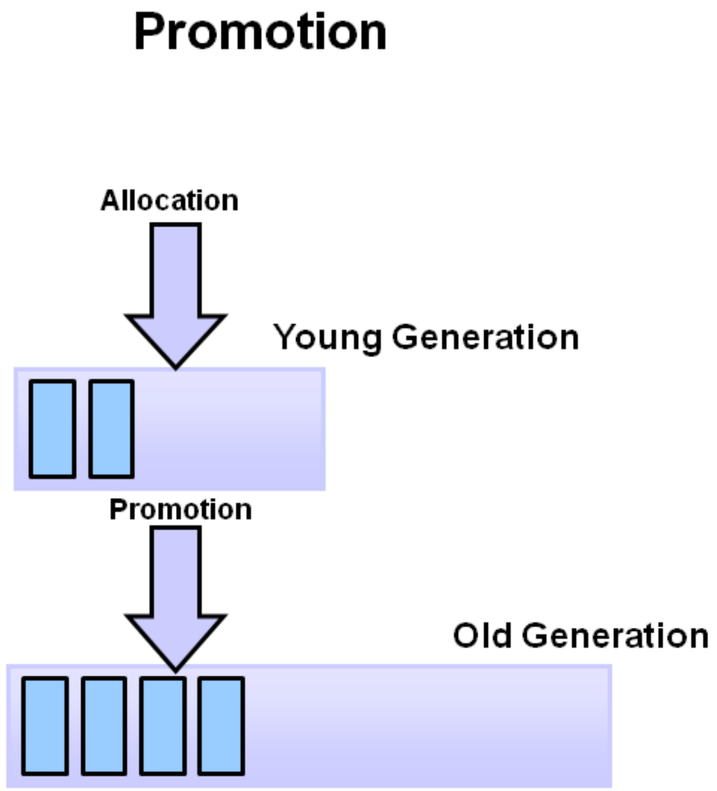
«Карьера» объекта в Java

6. Когда объект переживает достаточное кол-во minor-сборок, он становится достаточно старым, чтобы переехать в «старое» поколение.



«Карьера» объекта в Java

Обобщенная схема движения объекта.



OTUS

Как gc влияет на производительность?

- Потребляет системные ресурсы
- Stop the world

Какие бывают сборщики (Oracle Java 11)

Направления оптимизации сборщика:

- Throughput (Parallel Collector)
- Latency (CMS)

Наименование	Как включить
Serial Collector	-XX:+UseSerialGC
Parallel Collector	-XX:+UseParallelGC
CMS	-XX:+UseConcMarkSweepGC
G1	-XX:+UseG1GC
ZGC	-XX:+UnlockExperimentalVMOptions -XX:+UseZG

CMS collector is deprecated as of JDK 9.

ZGC is available as an experimental feature, starting with JDK 11.

Классический доклад Владимира Иванова.

OTUS

Демонстрация

Пример: GcDemo

OTUS

Мониторинг

Логирование работы gc:

-verbose:gc

-Xlog:gc*:file=./logs/gc_pid_%p.log

-XX:+HeapDumpOnOutOfMemoryError

-XX:HeapDumpPath=./logs/dump

OTUS

Мониторинг

Наблюдение за ходом работы:

Начиная с java 9 visualVM
не входит в сборку jdk.

Скачать можно отсюда.

Еще одна полезная утилита: jconsole

OTUS

Мониторинг

API для мониторинга в реальном времени:

java.lang.management.GarbageCollectorMXBean

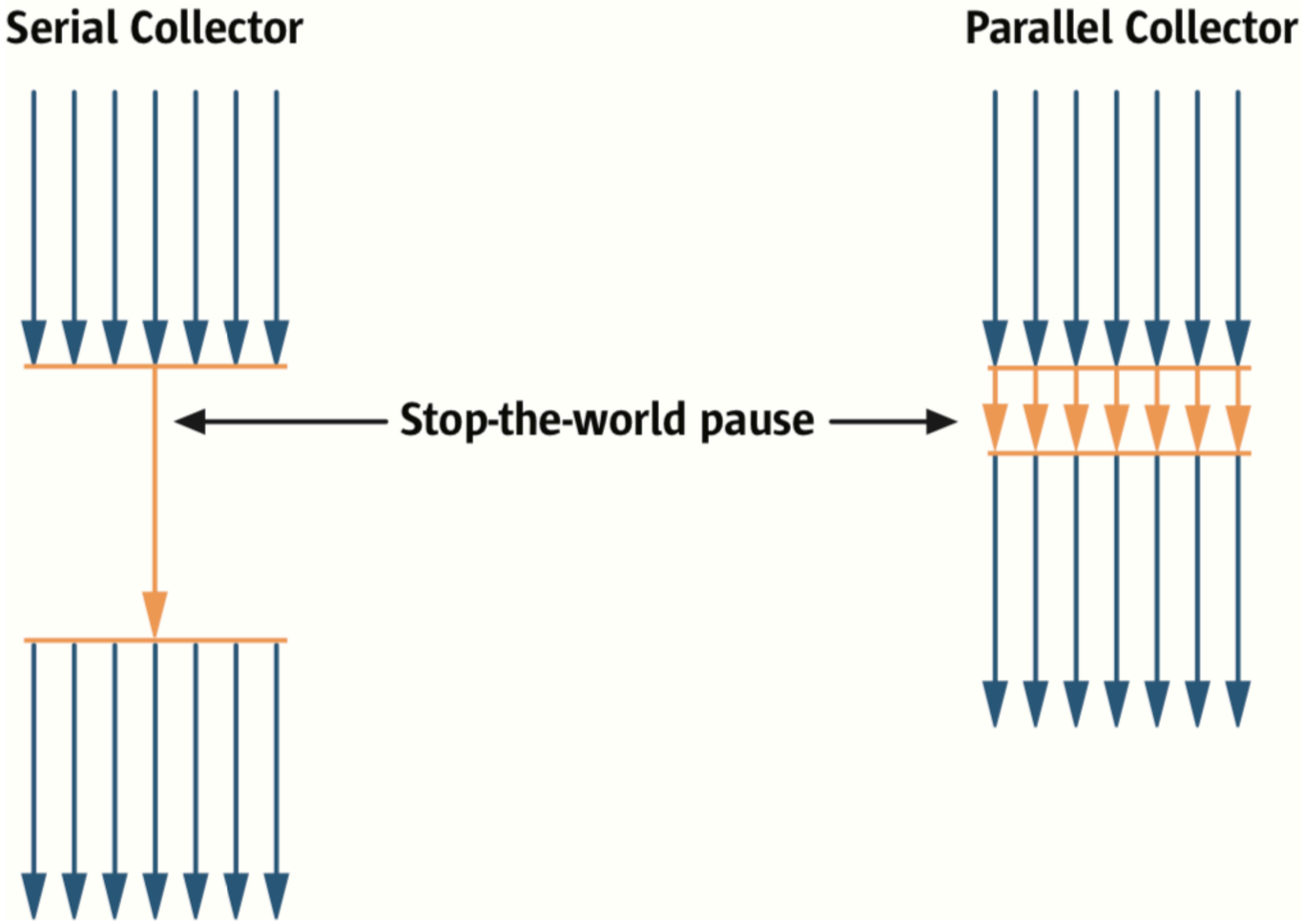
The management interface for the garbage collection of the Java virtual machine.

OTUS

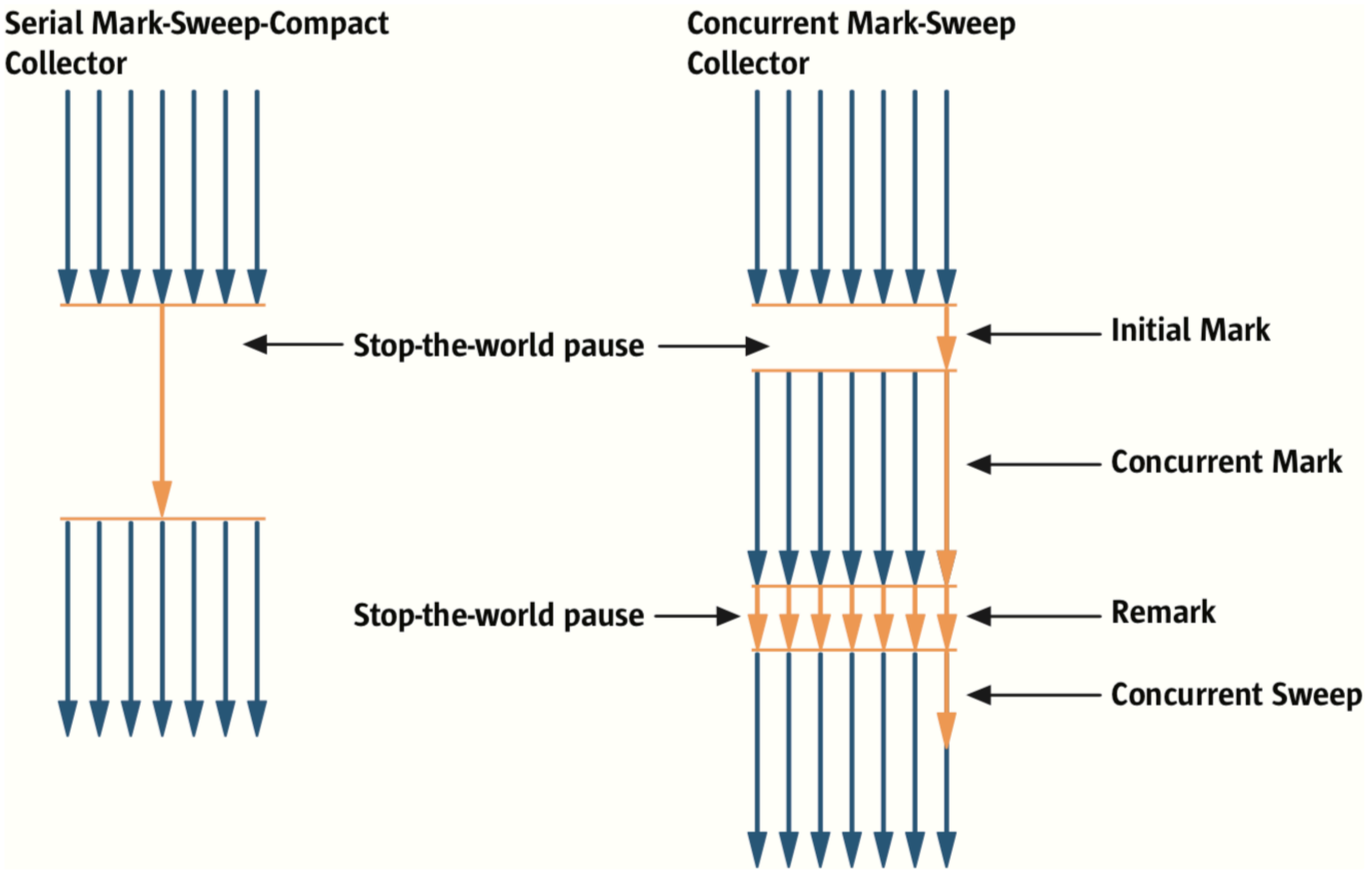
Еще один пример

Пример: Boxing

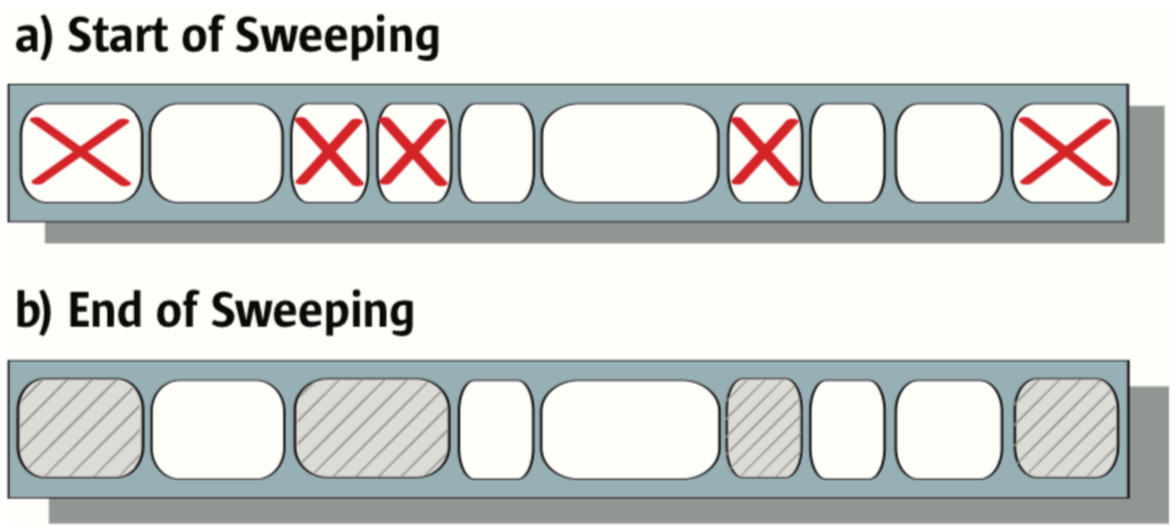
Serial & Parallel GC



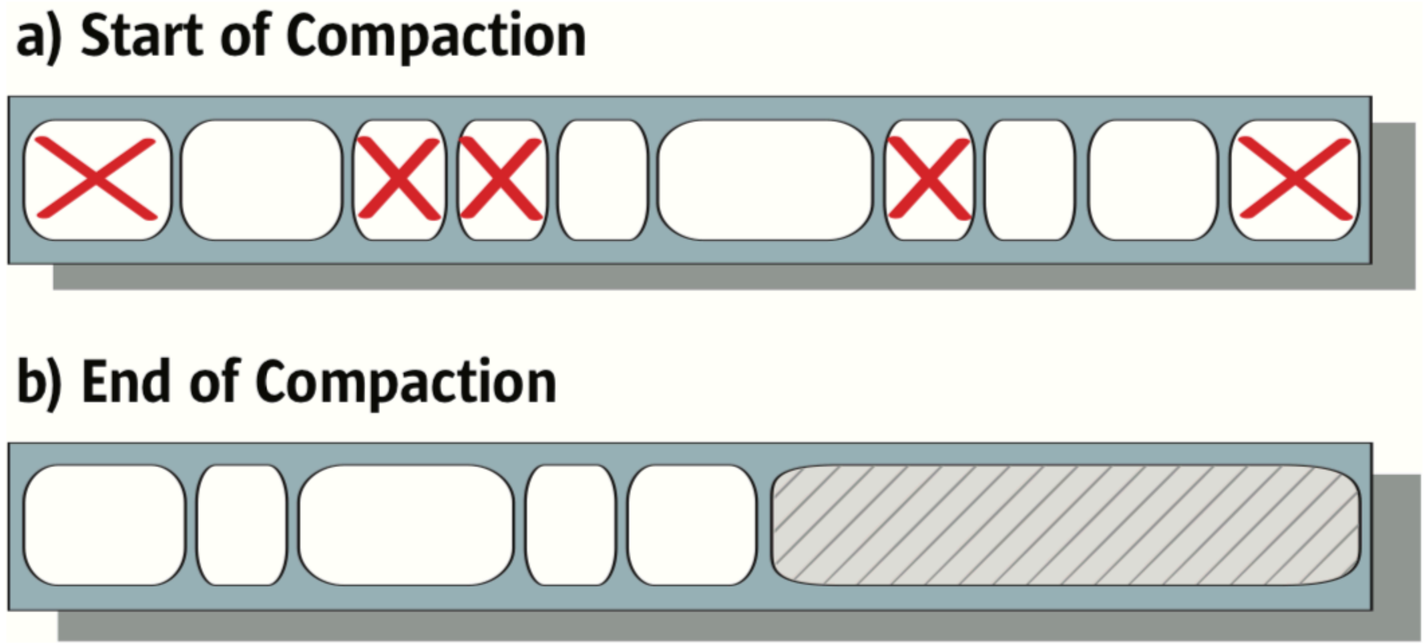
CMS


Serial Mark-Sweep-Compact Collector
Concurrent Mark-Sweep Collector
Stop-the-world pause
Initial Mark
Concurrent Mark
Stop-the-world pause
Remark
Concurrent Sweep

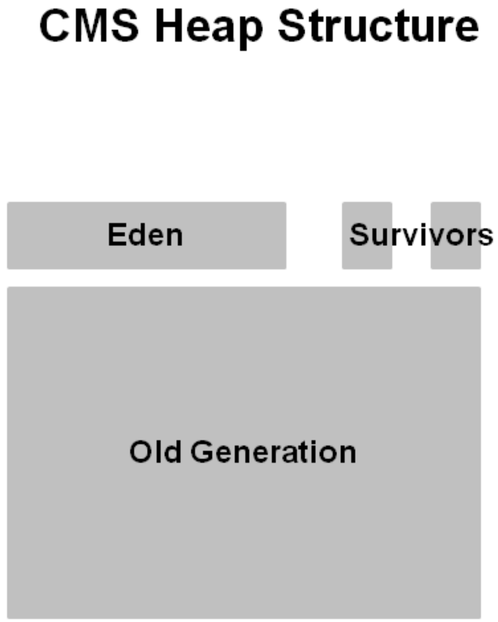
CMS



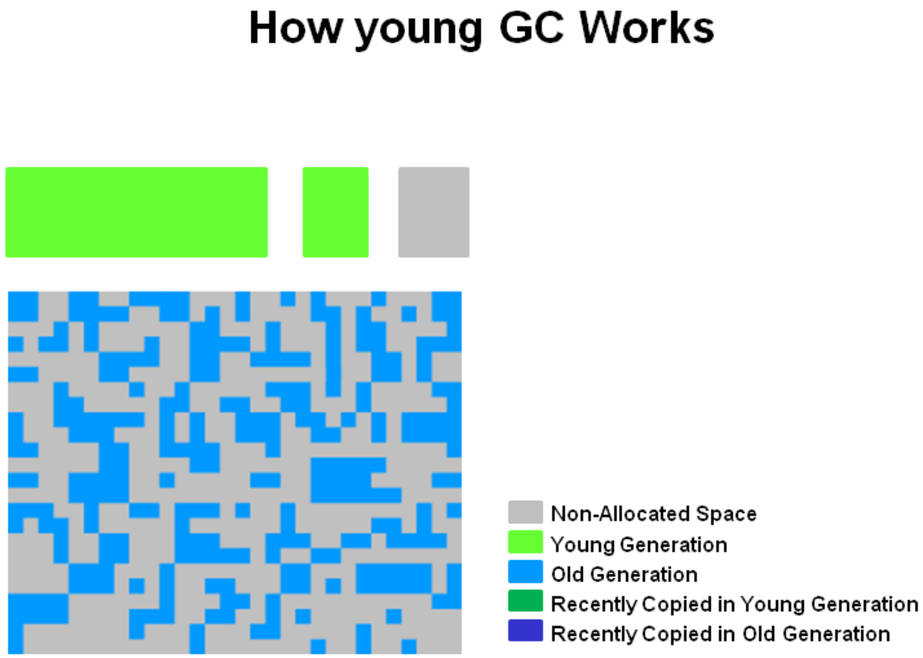
Compaction



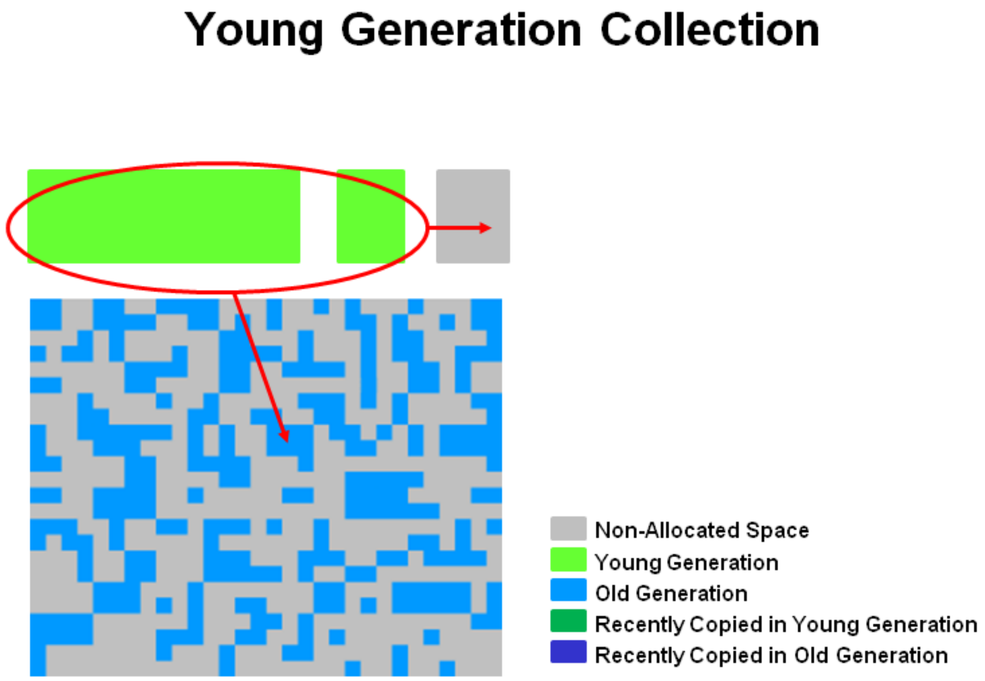
CMS



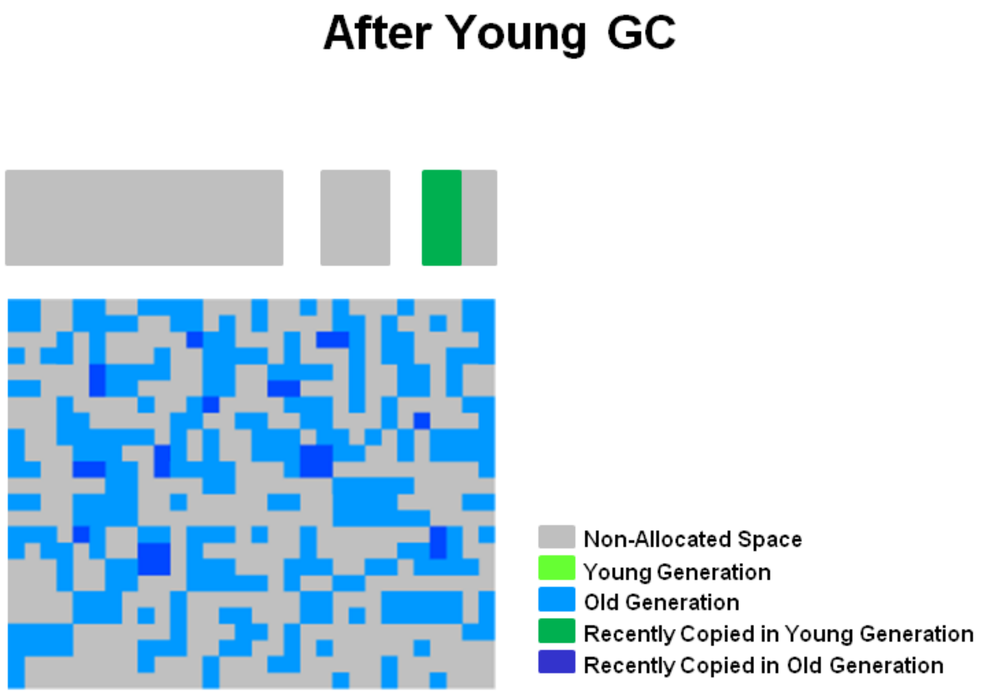
CMS


How young GC Works
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

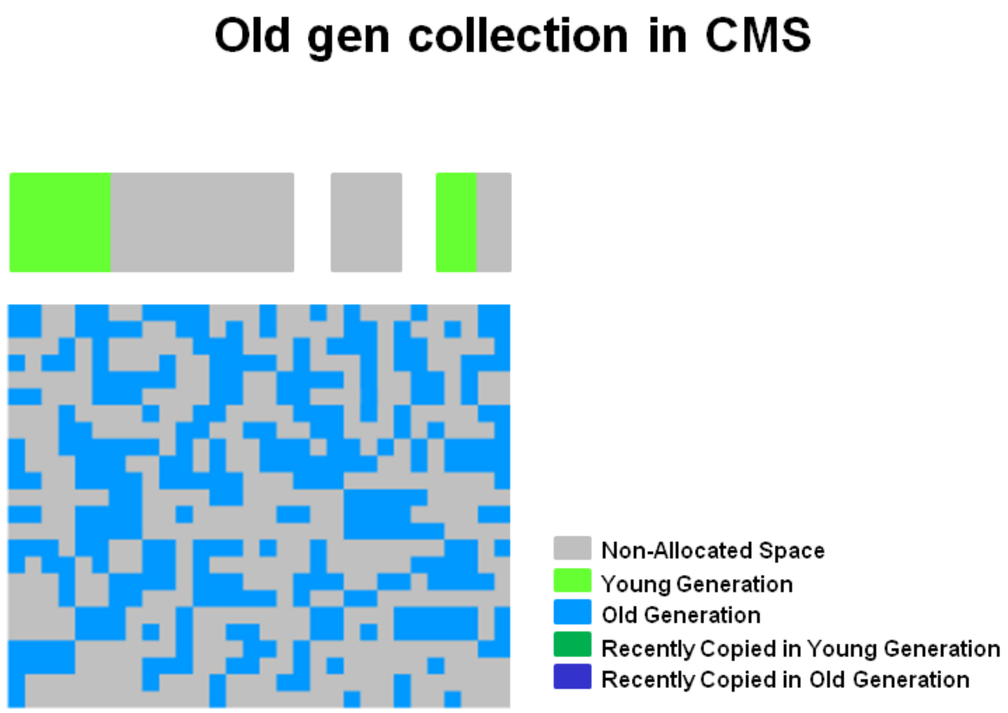
CMS


Young Generation Collection
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

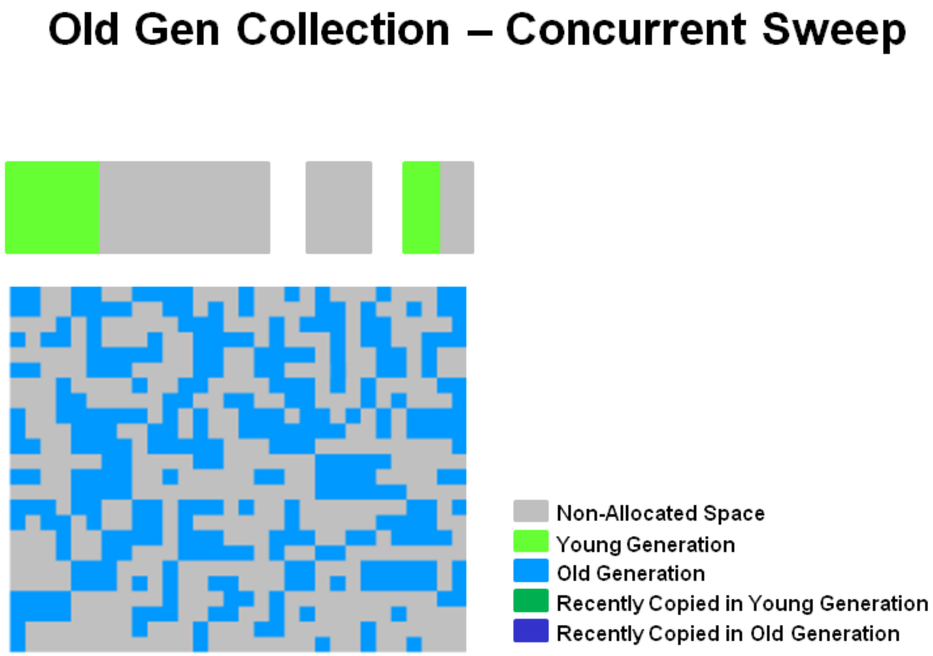
CMS


After Young GC
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

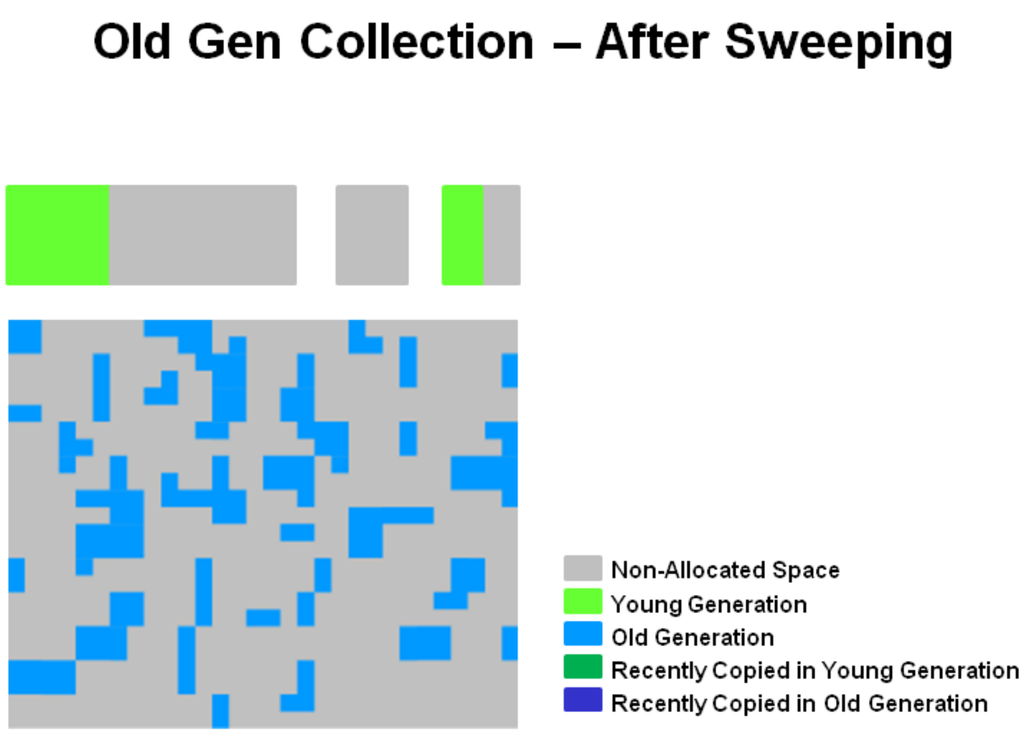
CMS


Old gen collection in CMS
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

CMS


Old Gen Collection – Concurrent Sweep
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

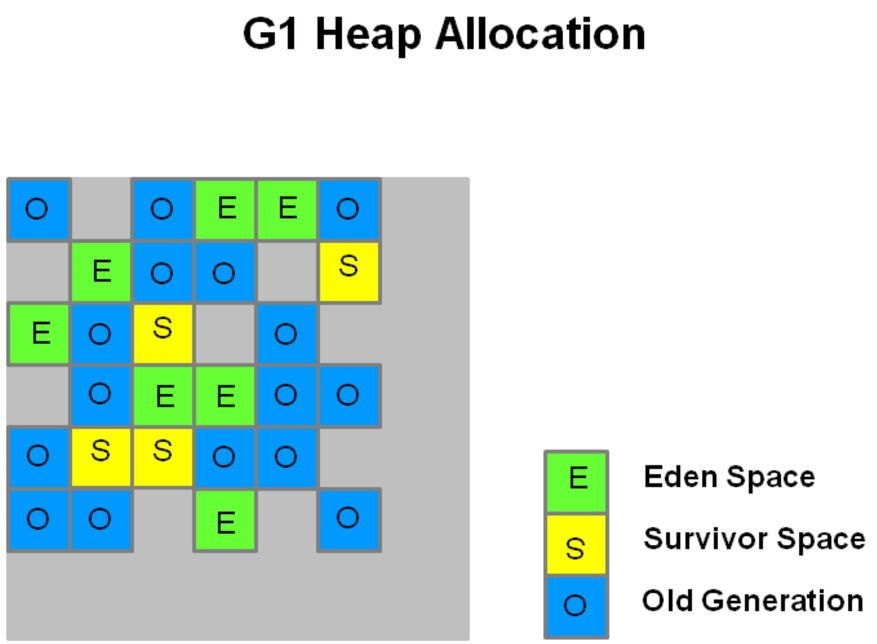
CMS


Old Gen Collection – After Sweeping
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

G1

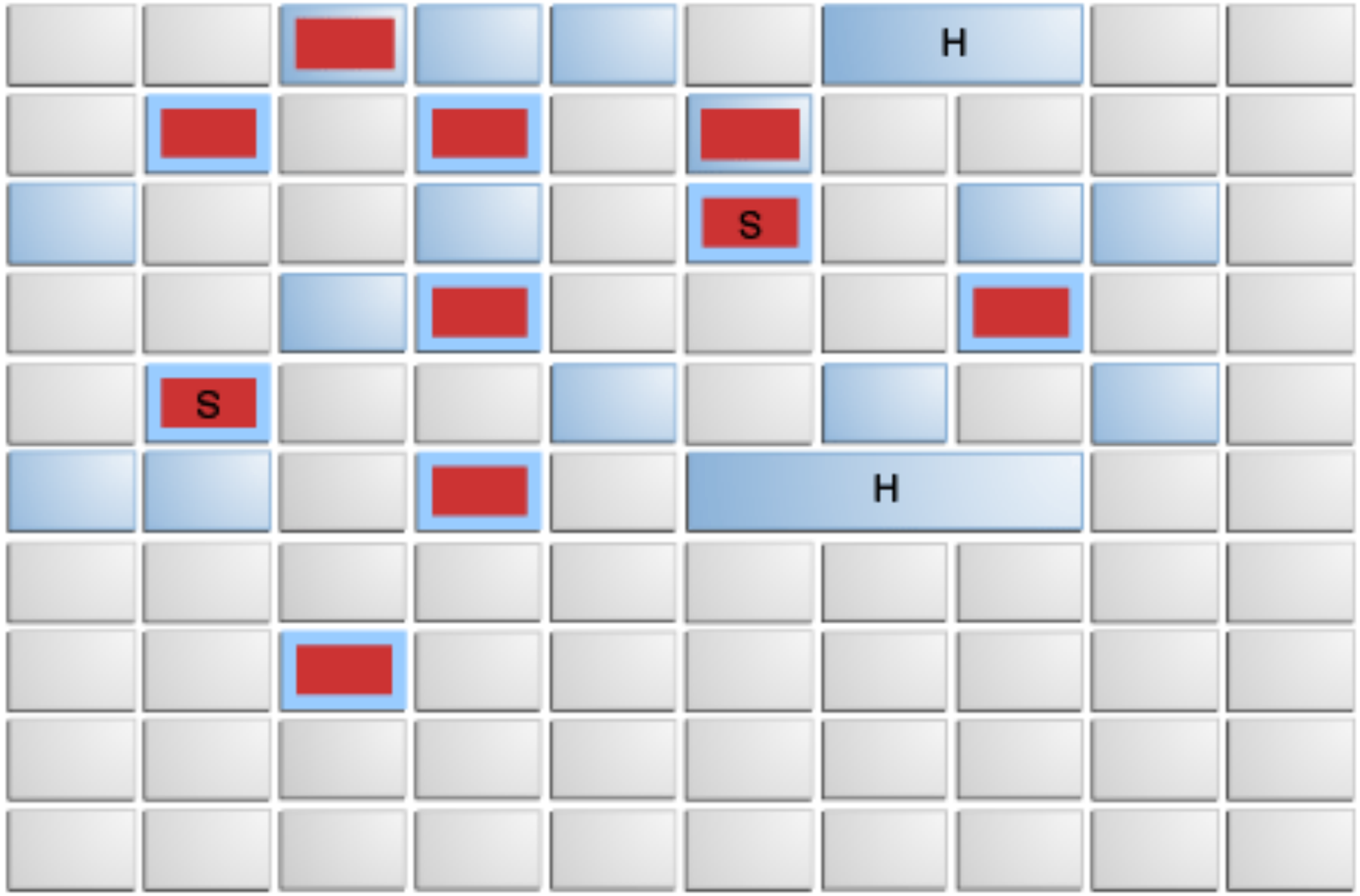

G1 Heap Structure
One memory area
split into many fixed
sized regions

G1

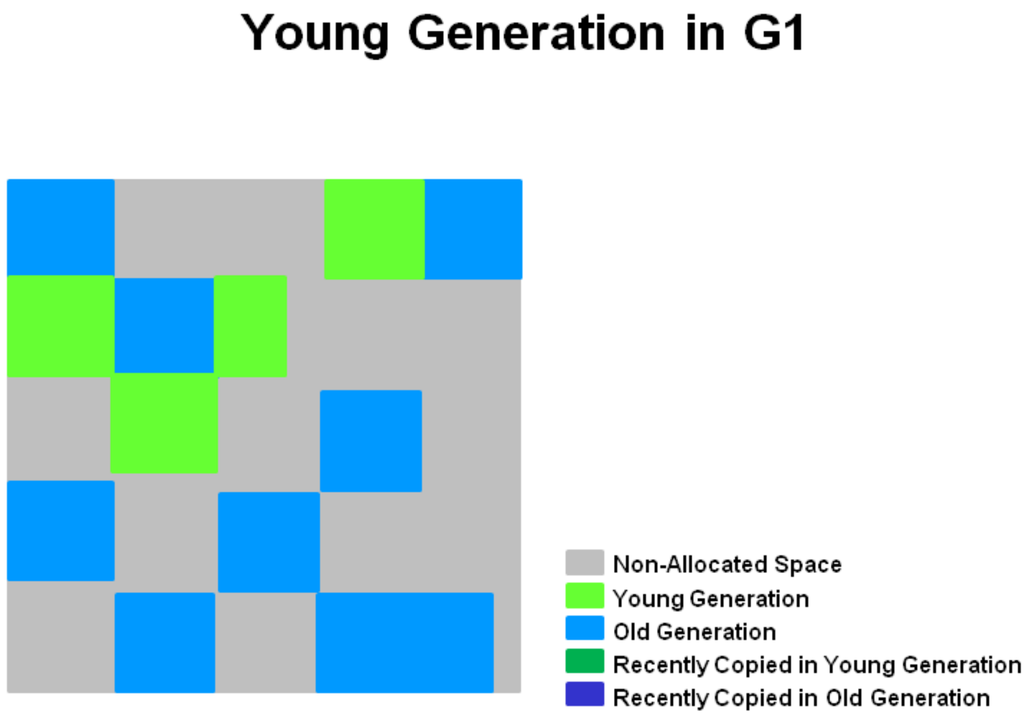

G1 Heap Allocation
O O E E O
E O O S
E O S O
O E E O O
O S S O O
O O E O
E Eden Space
S Survivor Space
O Old Generation

OTUS

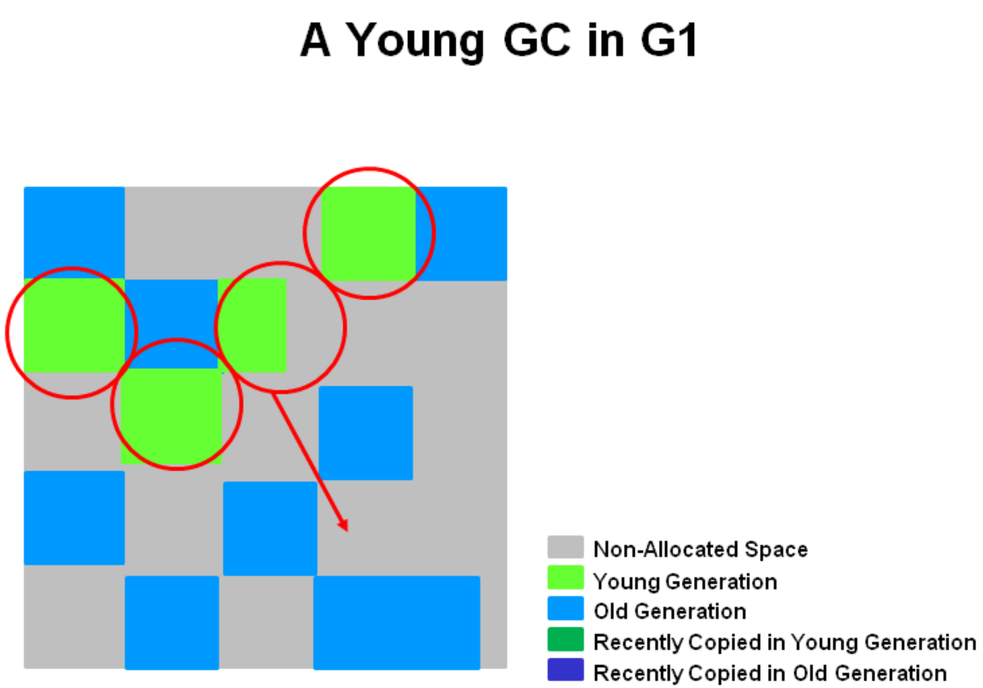
G1



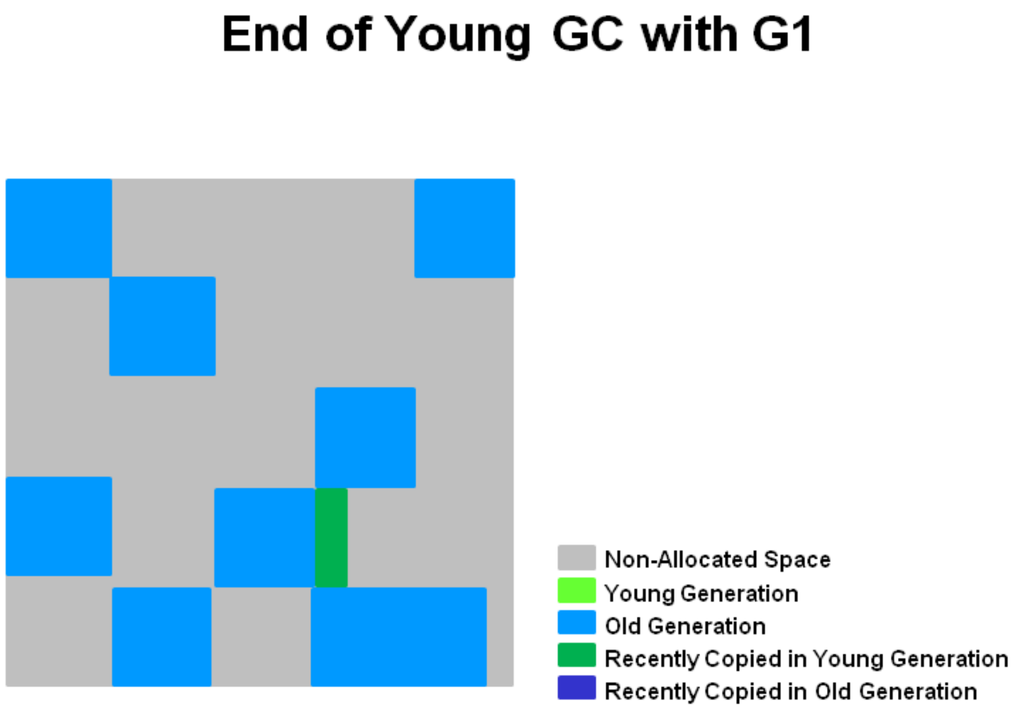
G1


Young Generation in G1
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

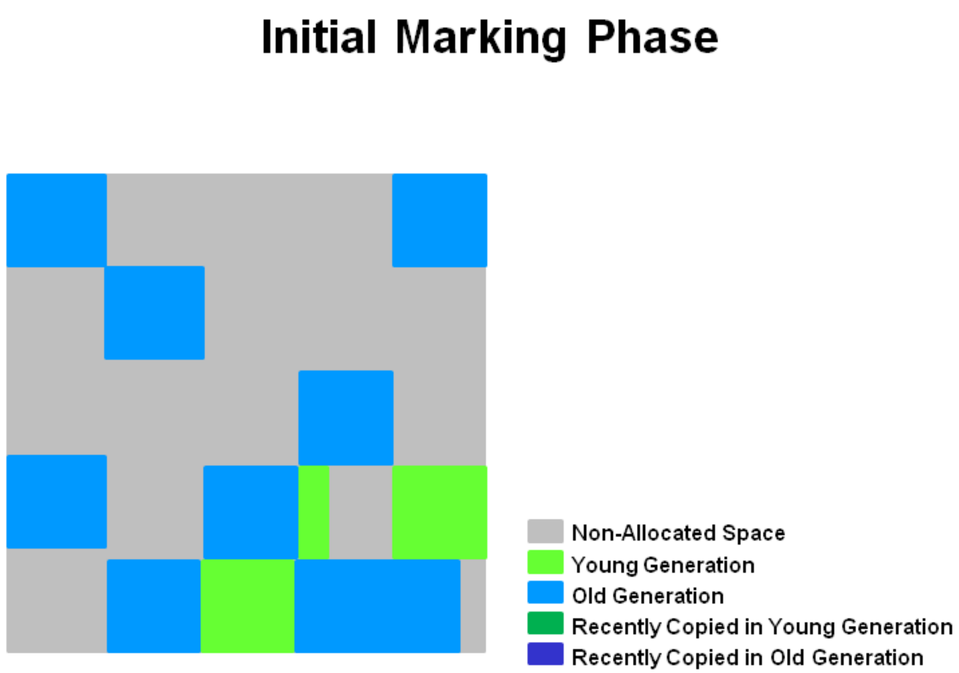
G1


A Young GC in G1
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

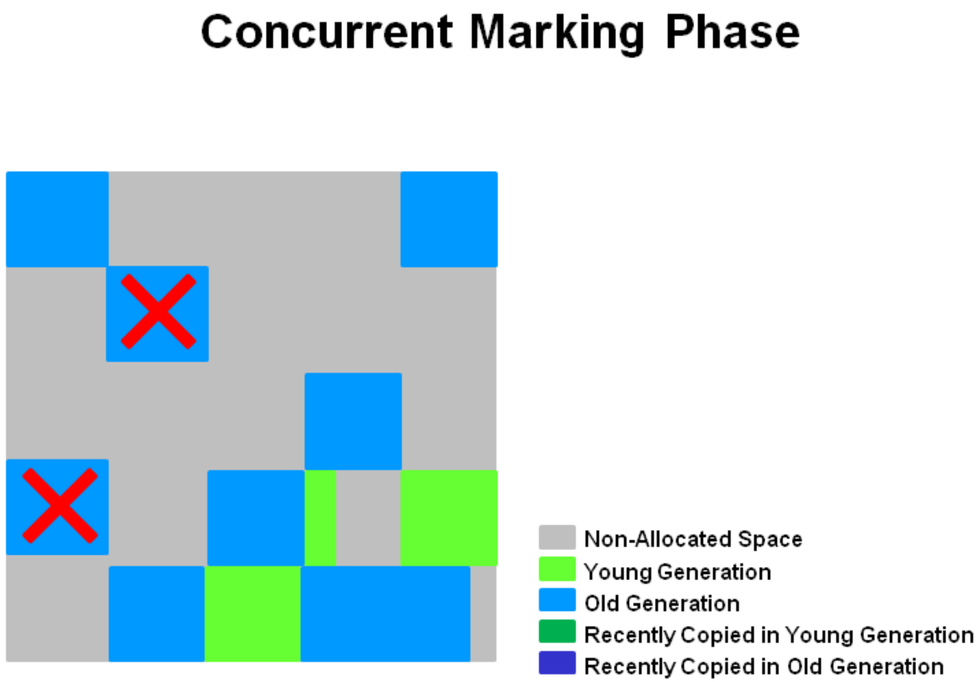
G1


End of Young GC with G1
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

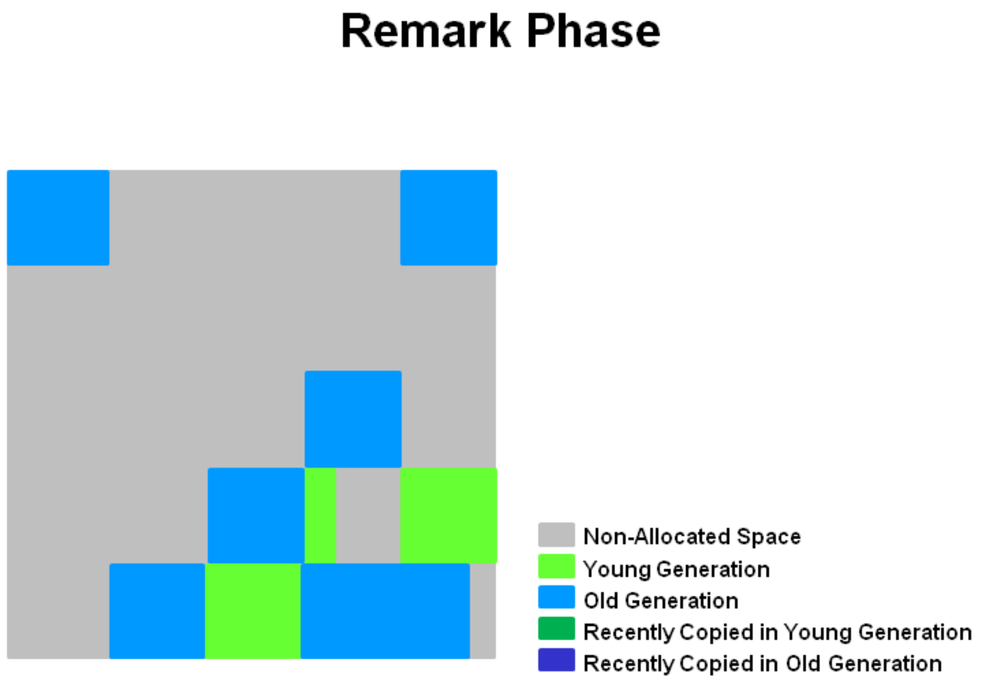
G1


Initial Marking Phase
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

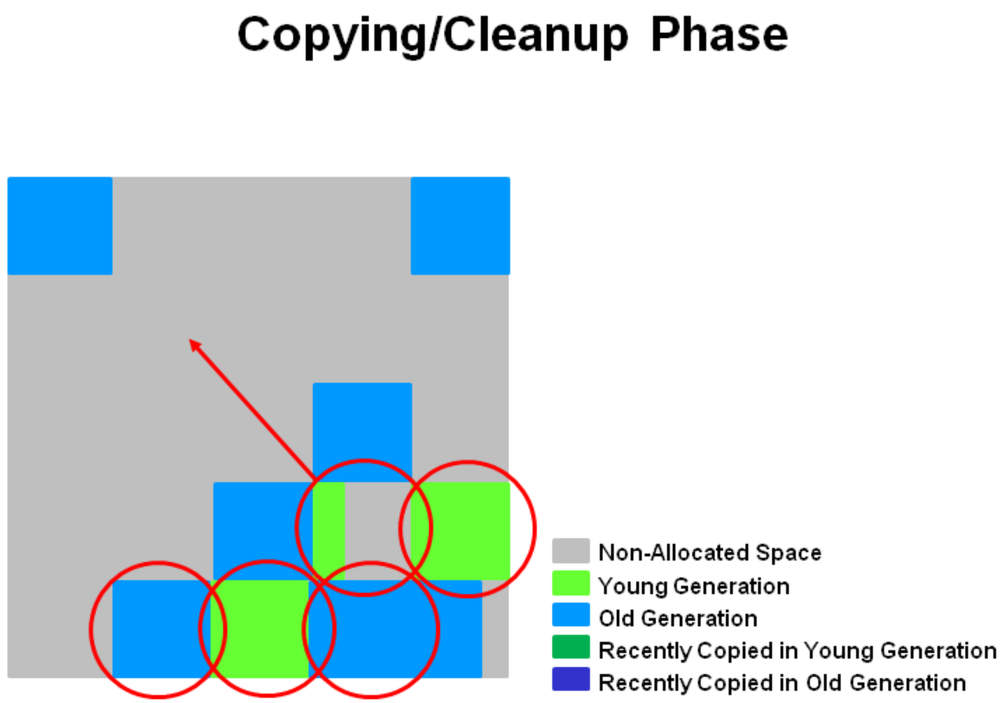
G1


Concurrent Marking Phase
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

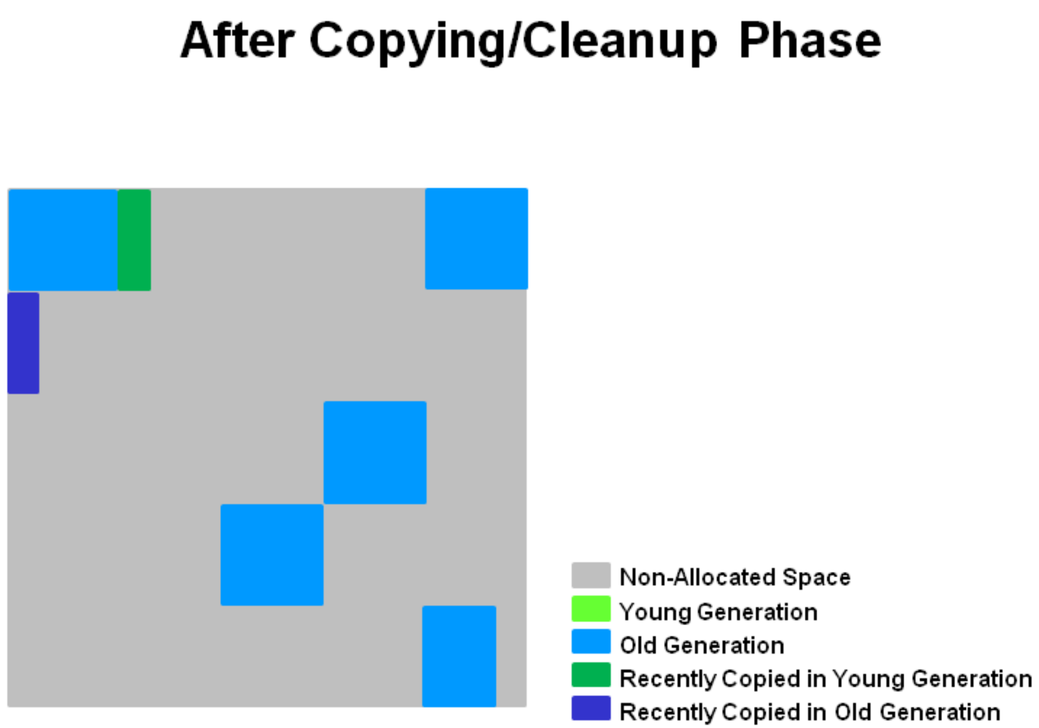
G1


Remark Phase
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

G1


Copying/Cleanup Phase
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

G1


After Copying/Cleanup Phase
Non-Allocated Space
Young Generation
Old Generation
Recently Copied in Young Generation
Recently Copied in Old Generation

Ваши вопросы?

Спасибо за внимание!