

Java内

1. 们 发人员 写 Java代

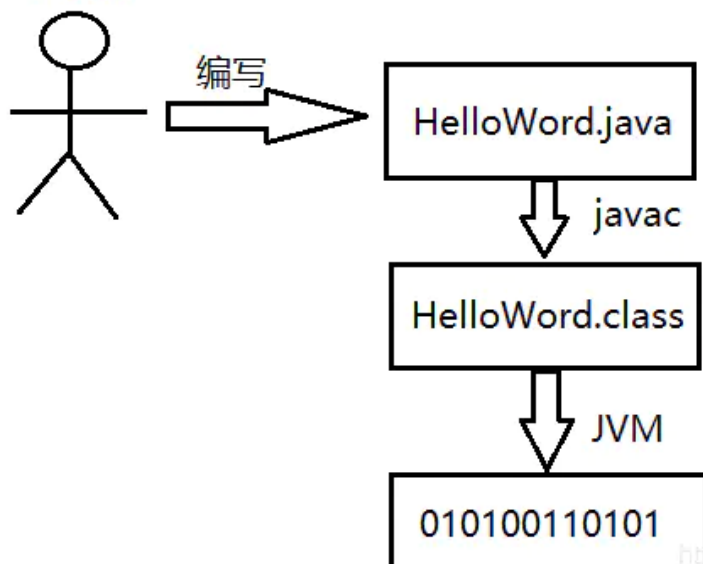
- 了 二
- He.. W d.ja a
- He.. W d.ja a

Java 件

1. .ja a
2. ja ac .c.a : 么 c.a 为JVM .c.a
3. JVM (于)

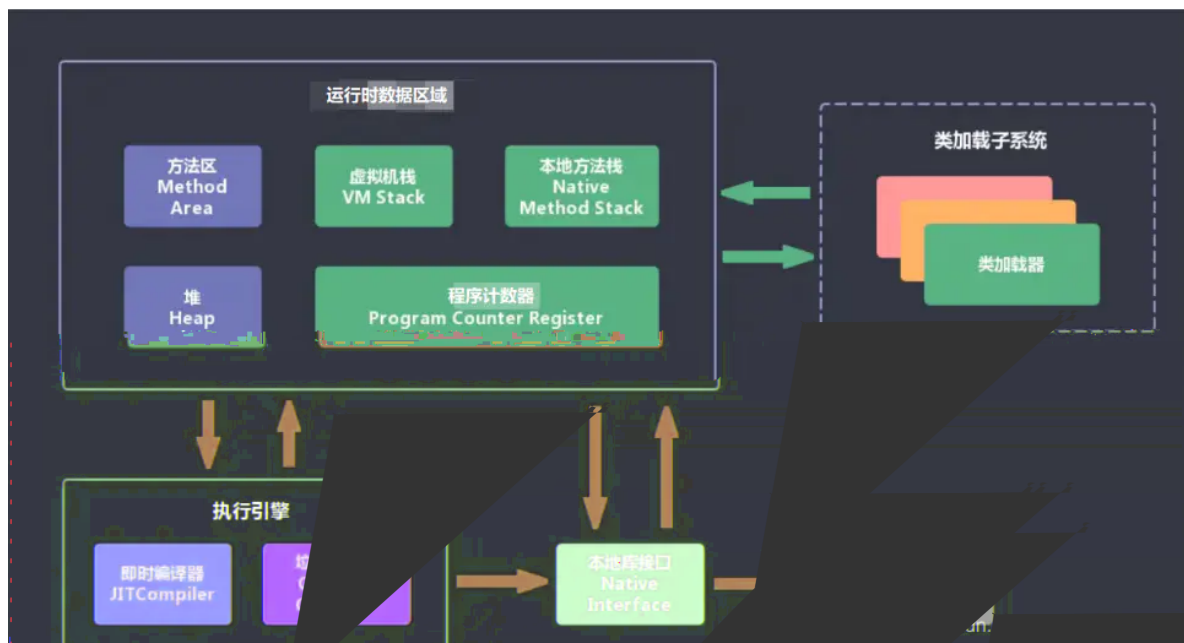
(是一个大概的观念 抽 画的概念)

程序员



2. 为什么 java 台

- 个 中 (JVM)
- Ja a JVM 了



- JVM 两个 两个 : 两个 为 C.a (C.a L o a d e r i n t e r f a c e ()) ; 两个 i e d a a a e a () N a i e l e f a c e () :
 - C.a () : () a . a g . O b j e c t ()
 - R i e d a a a e a 中 e h d a e a
 - E e c i e g i e () : c . a e
 - N a i e l e f a c e () : 与 a i e . l b a i e 互
 - R i e d a a a e a () : J V M
- : J a a , (C . a L o a d e r i n t e r f a c e) J V M

中, (R i e d a a a e a) , J V M

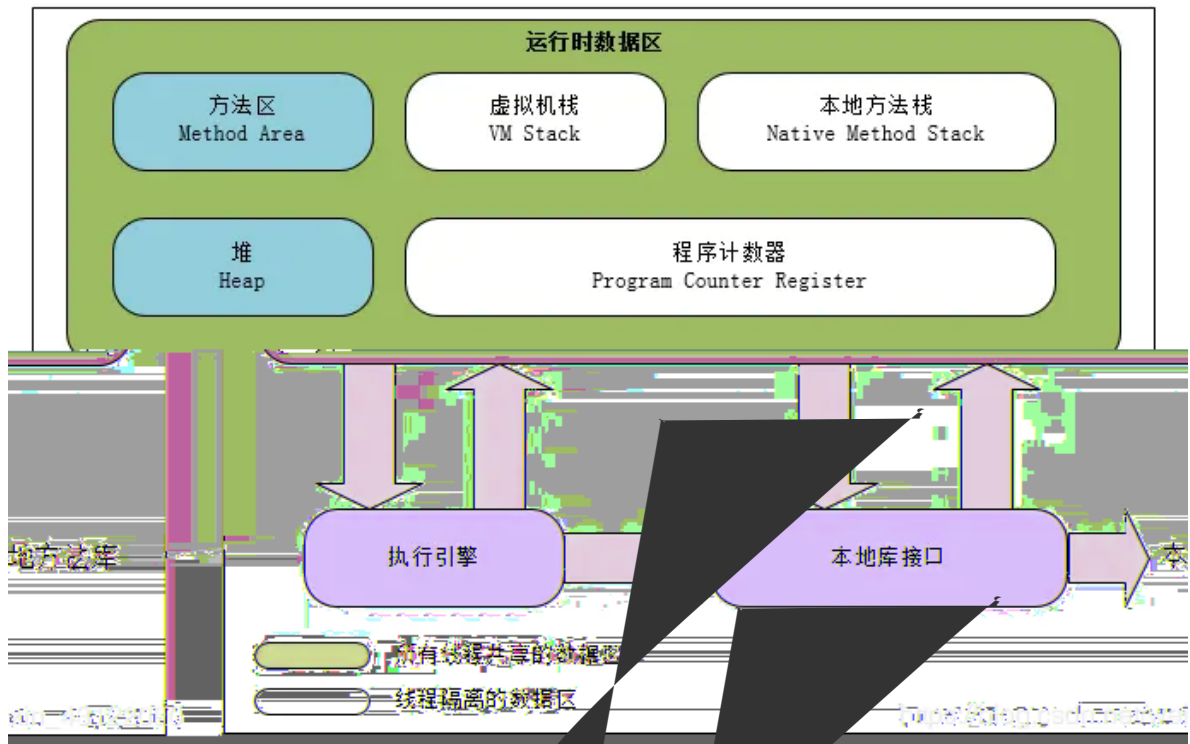
(E e c i e g i e) , CPU , 个 中

(N a i e l e f a c e) 个

5. 下JVM 区

- J a a J a a 中 为 个不 些 , 些 J a a 为 下 个 :

简单的 就是我们java 行时的东西是放在 的



- 程序计数器 (Program Counter Register) 是每个线程都有的，它记录了当前线程所执行的字节码指令的地址。如果当前线程正在执行一个方法，那么程序计数器就记录了该方法的入口地址。如果当前线程正在执行一个本地方法，那么程序计数器就记录了该本地方法的入口地址。程序计数器的作用是记录当前线程所执行的字节码指令的地址，以便在需要的时候能够快速找到该指令。程序计数器的值是由虚拟机自动管理的，程序员不需要关心它。
- 虚拟机栈 (Java Virtual Machine Stack) 是 Java 虚拟机中的一部分，它用于存放方法调用的栈帧。每个方法调用都会创建一个栈帧，栈帧中存放了方法的局部变量、操作数栈、动态链接、方法出口异常等信息。虚拟机栈的容量是有限的，如果栈帧过多，就会导致栈溢出 (Stack Overflow)。
- 本地方法栈 (Native Method Stack)：与虚拟机栈类似，但它是用于存放本地方法的栈帧。本地方法是指用 C 或 C++ 编写的代码，而不是 Java 代码。本地方法栈的容量也是有限的，如果本地方法调用过多，也会导致栈溢出。
- 堆 (Java Heap)：Java 虚拟机中用于存放对象的内存区域。堆的大小是由虚拟机自动管理的，程序员可以通过垃圾回收来管理堆内存。堆内存的分配和回收是由垃圾回收器负责的，程序员不需要手动管理堆内存。
- 方法区 (Method Area)：用于存放被虚拟机加载到的类信息、常量、静态变量、即时编译的字节码等。方法区是线程共享的，所有线程都可以访问方法区中的信息。

后有细的明JVM运行时数据区

6. 介绍一下？ ()

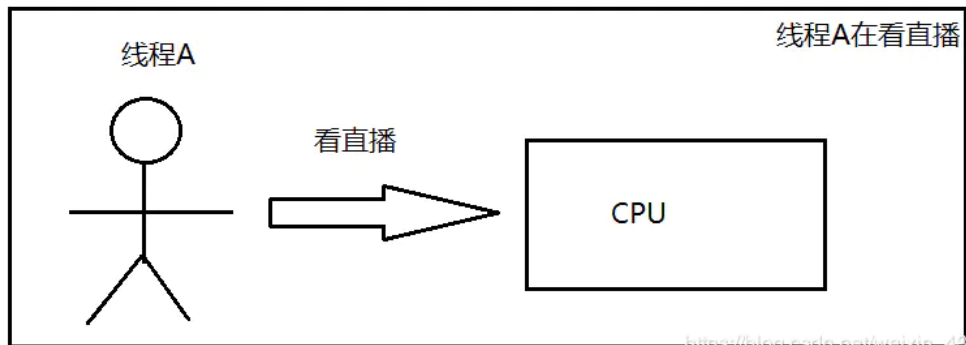
1. 虚拟机栈 () 是 Java 虚拟机中的一部分，它用于存放方法调用的栈帧。每个方法调用都会创建一个栈帧，栈帧中存放了方法的局部变量、操作数栈、动态链接、方法出口异常等信息。虚拟机栈的容量是有限的，如果栈帧过多，就会导致栈溢出 (Stack Overflow)。
2. 于Java 中，为了 个 个 之 互不 之为 O OfMe E

总结：也可以把它叫做线程计数器

- 例：Java中，CPU在CPU上运行，一个CPU上也可以运行多个线程。

- 假设：

- A



- B 了个 A, A

! [在 插入图片描] (https://user-gold-cdn.xitu.io/2020/4/13/171729fcc70da181?imageView2/0/w/1280/h/960/format/webp/ignore-error/1)

* 然后，视频结束后，线程A究竟干什么？（线程是最小的执行单位，他不具备记忆功能，他只去干，一个记忆就由：**程序计数器来记录**）

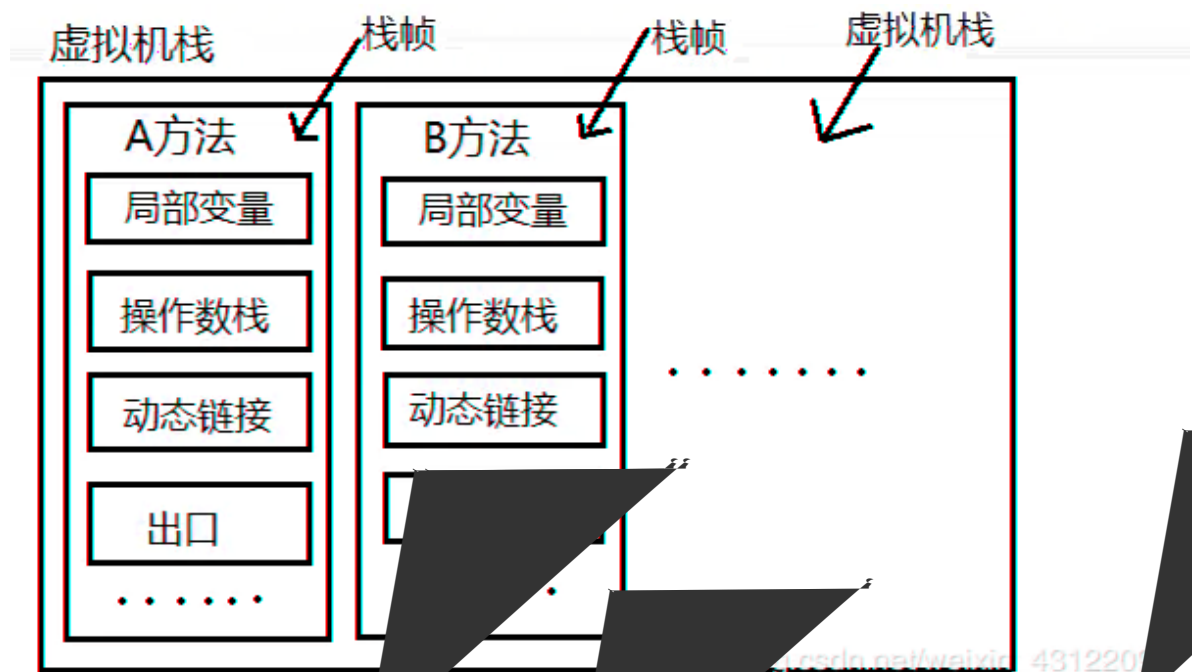
! [在 插入图片描] (https://user-gold-cdn.xitu.io/2020/4/13/171729fcc90c8a88?imageView2/0/w/1280/h/960/format/webp/ignore-error/1)

7. 介绍一下Java多线程？（多线程）

1. Java中，多线程是指一个程序中可以同时运行多个线程。

2. Java多线程的特点：每个方法在运行的同时，一个线程（Sack Face）于一个线程中。

- 多线程：一个程序中可以同时运行多个线程，一个线程中也可以运行多个线程。



1. 当方法A调用方法B时，在虚拟机栈中会创建一个新的栈帧，这个栈帧中会包含方法B的局部变量、操作数栈、动态链接和出口。当方法B执行完毕并返回时，这个栈帧会被销毁，虚拟机栈中只剩下方法A的栈帧。
2. 在虚拟机栈中，每个栈帧的大小是固定的，通常为8个字节。如果方法B的局部变量或操作数栈中的数据超过了这个限制，就会发生栈溢出错误。
3. 虚拟机栈的容量是有限的，可以通过设置 JVM 参数来调整。如果栈容量不足，可以通过增加栈容量来解决。
4. 虚拟机栈的垃圾回收是由 JVM 自动管理的，不需要程序员手动干预。

那么，如果方法A调用方法B，会创建新的栈帧吗？

- 是的，当方法A调用方法B时，在虚拟机栈中会创建一个新的栈帧，这个栈帧中会包含方法B的局部变量、操作数栈、动态链接和出口。

向什么方向生长？

- 虚拟机栈的容量是有限的，可以通过设置 JVM 参数来调整。如果栈容量不足，可以通过增加栈容量来解决。

会创建新的栈帧吗？

- 是的，当方法A调用方法B时，在虚拟机栈中会创建一个新的栈帧，这个栈帧中会包含方法B的局部变量、操作数栈、动态链接和出口。

8. 你了解 Java 吗? ()

- Java 是一种面向对象的编程语言，它是由 Sun Microsystems 公司开发的。Java 的语法与 C++ 类似，但它具有更强的安全性和可移植性。
- Java 的虚拟机（JVM）是 Java 程序运行的环境。JVM 负责将 Java 字节码编译成机器码，并在目标平台上运行。
- Java 的垃圾回收（GC）是由 JVM 自动管理的，不需要程序员手动干预。GC 负责回收不再使用的内存，防止内存泄漏。
- Java 的类加载器（ClassLoader）负责将 Java 类加载到 JVM 中。类加载器会根据类名找到类的字节码，并将其加载到内存中。
- Java 的字节码（.class 文件）是 Java 程序的中间表示形式。字节码可以在任何支持 JVM 的平台上运行，这保证了 Java 的“一次编写，到处运行”的特性。

- `Ja a` (`-X` `-X` `O OfMe E` , `ja a` 于 上不 中 主 , 且 也

9. 不 下 ?

1. , , 不 上 了 `a i e`
2. 为 `Ja a` (也)
3. `a i e` 不 , `ac.e` 下 , 且 `a i e`
4. `C C++` 中 `C C++`

10. 不 下 区 ()

1. , 于 `Ja a`
2. 个 `N -Hea` () , `O OfMe E`

11. 什么 JVM

- , ,
- 个 于 , 不 上 , JVM

12. 你听 内 吗?

- (`Di ec Me`) 不 , 也不 `Ja a` 中 义 , 也 `O OfMe E` ,
- 于 `Ja a` 中

13. 吗?

- 中, (些 于 , 为 些 , 不 了 , 亡), 为了 , `ja a` 中不 (GC)
- `Ja a` , 也 不 , `Ja a` ,
- 为 `Ja a` `Ja a`

14. 区别 什么?

| | JVM | JVM |
|--|--------------------|-----|
| | GC 也 不 , 不 些 (,) | 中 , |
| | 为 不 , 于 | , , |
| | | : , |
| | 于 个 | 于 也 |

- :
-
-

15. 和

- (hashtable) 了 个 ,
- (default) 了 个 且 了 个 , 个 个
- :
- : 中 内 于

16. Java会 内 吗? 为什么?

- 不 中 上 , Java GC , 也 , 不 , GC , 中
- , , Java也 , Java : , 尽管短生命周期对 已经不再 要, 但 是因为 生命周期对 持有它的引用而导致不能被回收, Java中

制及

17. Java 制

- java中, 不 个 , JVM中, 个 , 下 不 , 不 , 些 , 中,

18. GC 什么? 为什么 GC

- GC (Garbage Collection), 不 , java GC , java

19. 优 和

- : JVM 不 了, 个
- : 不 个

20. 原 什么? 办 动 ?

- 于GC , , GC 个
- , GC (heap)中 些 "
- " , 些 "不 " GC 些 为"不 " , GC 些
- (System.gc()), GC , java 不 GC

21. JVM 中 哪些 ?

- : gc 不
- : 不 , 之
- : 不 , 下 GC
- (/) : , Phase Reference , gc 个

22. 么判 否可以 ?

- , 不 ; 些 , 些
- 两 :
 - : 为 个 个 , +1, -1, 为0 个 不 ; (个 了)

- GC R 下 , 为 个
- GC R (上)

23. Full GC 什么

- 个 久
- 为F .. GC 个 F .. GC , Ja a 中 F .. GC

24. 什么 候可以

- 个 不 , 个 了
- 不 久 , 久 了 了临 , (F .. GC) 久 也 为 么 久 F .. GC

25. JVM 哪些?

- - : , : 不 ,
- : 二个 , : 不 , 上,
- - : , ,
- : 不 为 , ,
-
- ,
- - (Ma k-S ee) , 为两个 :
- :
- :
- - 之 , 为 上
- 优 : , 不
- : , 不 , 了
- - 下

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

存活对象

可回收

未使用

https://blog.csdn.net/wandngj3122090

制

- 为了 - 不 , 了 为两个 , 中 个 , 中,
- 优 : , 不
- : 为 ,
- 下

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

存活对象

可回收

未使用

保留区域

https://blog.csdn.net/wandngj3122090

-
- 中 , 不 了, 为 , 中, (Ma k- C ac) , 与 - 不 , ,
- 优 : 了 -
- : 上 了
- - 下

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |

存活对象

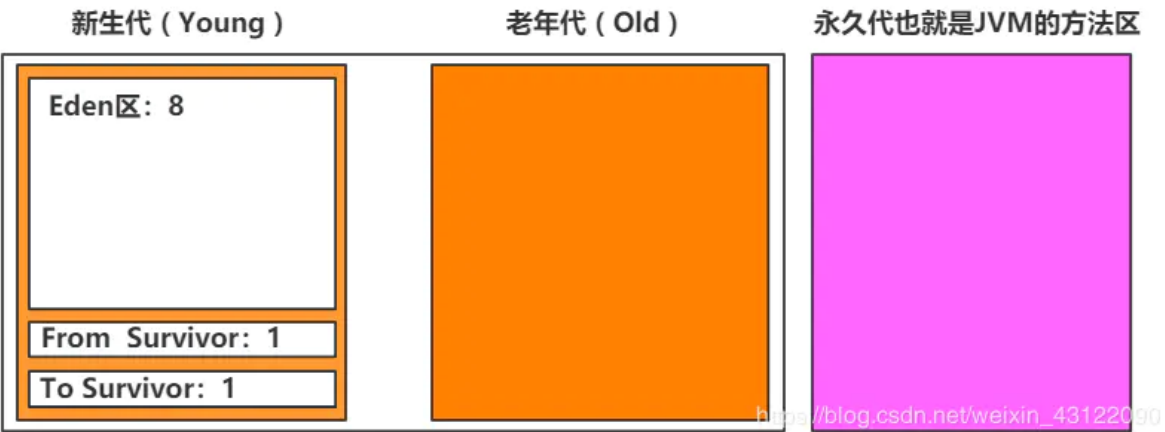
可回收

未使用

https://blog.csdn.net/weixin_43122090

分代

- 业界将垃圾回收分为新生代、老年代、永久代，其生命周期如下（后有点讲解）



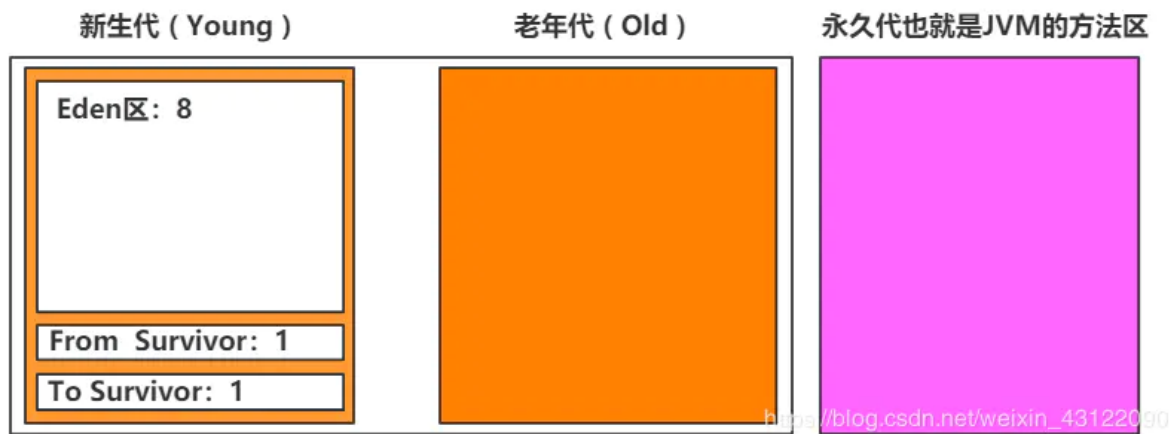
https://blog.csdn.net/weixin_43122090

26. JVM中 老年代中会发生什么

- 在老年代中，垃圾回收器（Full GC）会进行垃圾回收，将垃圾对象回收，并将垃圾对象放入垃圾回收池中（如：Java 8中的垃圾回收池，如：a1e）。

以及 代 代 永久代

27. 新生代 老年代 永久代 区别



- 在Java中，有两个不同的垃圾回收器：Young (Y) 和 Old (O.d) (Y) 和 (O.d) 为了 JVM 中，
- 在 Young 中，使用了复制算法，
- 在 Old 中，使用了久，
- “标记-清理”或者“标记-整理”
- 在 JVM 中，有些

28. Minor GC Major GC Full GC 什么

1. Minor GC 是 GC, 于Java
2. Major GC 是 GC, Maj GC 是 Mi GC
3. Full GC 是个

29. Minor GC Major GC Full GC区别及 发 件

- Minor GC 发 件 为
 1. eden 区, Major GC 个, eden 不,
 2. Major GC 是 Mi GC
- Major GC和Full GC 发 件 为: Major GC 常是 full GC是等价的
 1. >
 2. Minor GC 了
 3. 久
 4. Survivor (S)
 5. CMS GC
 - 6.

30. 为什么 代 分Eden和两个 Survivor 区 ?

- Survivor 区, Eden 区, Minor GC, Major GC. 于, Full GC 是 Mi GC

- S i 为Ede S i 义, 15 Mi GC 中
- 两个S i 了 GC, Ede 中 Mi GC, Ede S0中 S1 (个 , 为)



31. Java 代(Old)和 代(Young) 例?

- (Y g)与 (O.d) 为 1:2 (XX:Ne Ra i), : (Y g) = 1/3 (O.d) = 2/3
- 中, (Y g) 为Ede **两个 Survivor 区**, Ede 个S i = 8:1:1 (XX:S i Ra i),
- JVM Ede 中 S i 为 , 么 , S i

32.为什么 分代:

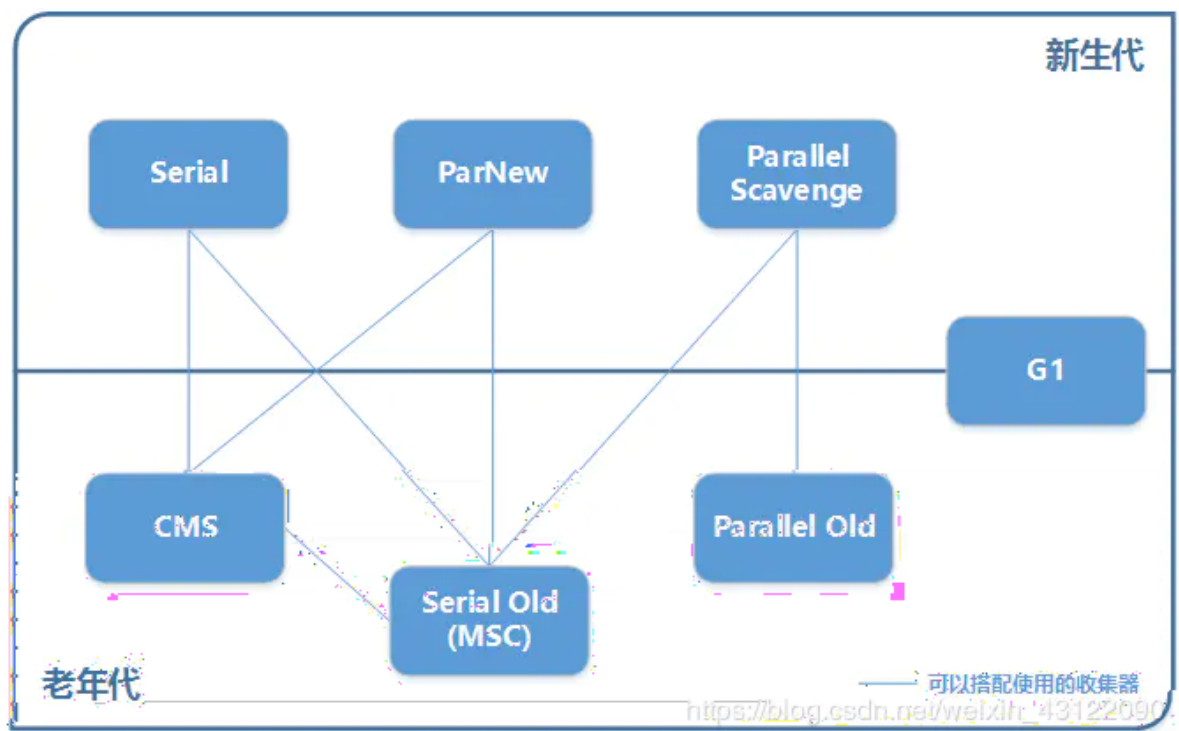
- 主 个 , 于 , :
 - 中, , 了 ,
 - 中 为 , -
- 为Ede S i (F 与T , 个) 两个 上 三个 Ede 中 (也 , 么 (ja a) Ede j Mi GC, Mi -GC , 且 S i , 么 S i 中 为1, S i Mi GC, 1, (为15) , 中了,

33. 什么 他和 什么区别

- () , 不 JVM

34. 下JVM 哪些 ?

- 7 于不 , 中 于 Se ia. P aNe Pa a.e. Sca e ge, Se ia.O.d Pa a.e.O.d CMS, 于 个Ja a G1 不 之



| | 作 区 | | 作 | | |
|----------------------|--------|--------|---|--|--------------------------|
| Se ia. | | | | | C.ie 下 |
| Pa Ne | | | | | Se ia. , Se e 下 , CMS |
| Pa a..e. Sca e ge | | | | | |
| Se ia. O.d | | - | | | Se ia. , C.ie 下 |
| Pa a..e. O.d | | - | | | Pa a..e. Sca e ge , |
| G1 | + | - + | | | JDK1.9 |

- Se ia. (): , , ;
- Pa Ne (): , 上 Se ia. , CPU
下 Se ia. ;
- Pa a..e. Sca e ge (): , CPU
= /(+GC), CPU ,

- Serial Old (-) ;
- Parallel Old (-) : , , Parallel Scavenge ;
- CMS (Concurrent Mark Sweep) (-) : , GC 为
- G1 (Garbage First) (标记整理 + 复制算法来回收垃圾) : Java , G1 JDK1.7 个 , G1 于 - , 也 不 , G1 不 于之 个 : G1 个 Java (,) , 于

35. 可以 么分 ? (了 了)

```
Serial / Serial Old
Serial / CMS
ParNew / Serial Old
ParNew / CMS
Parallel Scavenge / Serial Old
Parallel Scavenge / Parallel Old
G1
```

36. 代 和 代 哪些? 什么区别?

- : Serial, ParNew, Parallel Scavenge
- : Serial Old, Parallel Old, CMS
- : G1

37. 分代 么 作 ?

- 两个 : , 1/3, 2/3
- , 3个 : Eden, Tenured, From Space, 8:1:1, 下:
 - Eden + From Space Tenured ;
 - Eden From Space ;
 - From Space Tenured , From Space Tenured , Tenured From Space
- From Space Tenured , +1, 15 (15) , 为 也
- 个 之 , 上 些 了 个

内 分

38. java内存分与以及Minor GC和Major GC

- 内存分与，也两个了
- 内存分与，Ja a 上 (，些下也上，TLAB上主Ede，了，下也上，下也于，下世：

39. 优先 Eden 区分

- 内存分与，Mi GC，Ede GC，Ede，中
 - Mi GC，GC，中 Maj GC/Full GC
 - **Minor GC** GC，为Ja a，Mi GC也；
 - **Major GC/Full GC** GC，了Maj GC，Mi GC Maj GC Mi GC 10 上

40. 为什么入代

- 内存分与，GC，不
- 内存分与，Ede 两个S i 之，于

41. 入代

- 些, 么 些, Ede
个 义了 个
S i S i 中 为1
S i 中 Mi GC (15)

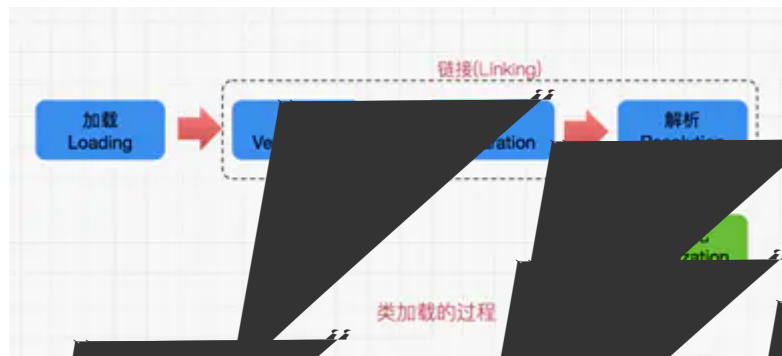
加制

42. java 加制?

- C.a
ja a

43. 加制及

- 主 个, 中, JVM 3个
JVM 3个, 也 个3个
为



1 加

- 中 些 中 个
ja a..a g.C.a 个
Ja a JVM, 为
C.a L : a C.a
C.a 不 二 : a C.a
中 于 中 C.a 不), 了
中, 且 了, Ja a

2

- 之, 为之 个 C.a
二 JRE中 (ja a 二 JVM 之中)
为 下3个

1. : JVM , 主 C.a
2. : 为 (a) , 些
3. : 为

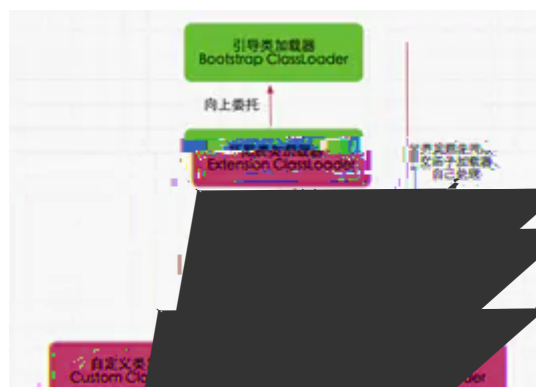
3 初 化

- `<clinit>()` `<clinit>()`
 - 中 所有类变 的 (static块)中的 句合并产生, 代码从上往下执行。
 - 个
 - 个 `<clinit>()` 中
- 初始化的总结就是: 初始化是为 静态变 予正确的初始值

44. 下JVM加 Class 作

- Ja a中 c.a 中 , 乎不 , 为 些
- , 两 :
 1. , e ,
 2. , c.a .e() ,
- Ja a , 不 , 为了 () j 中, 于 ,

45. 什么 加 , 加 哪些?



-
- 主 下
 1. (B) Ja a Ja a
 2. (e) i Ja a Ja a
 3. () Ja a (CLASSPATH)
Ja a , Ja a
C.a L ge S e L
 4. 义 , ja a.a g.C.a L

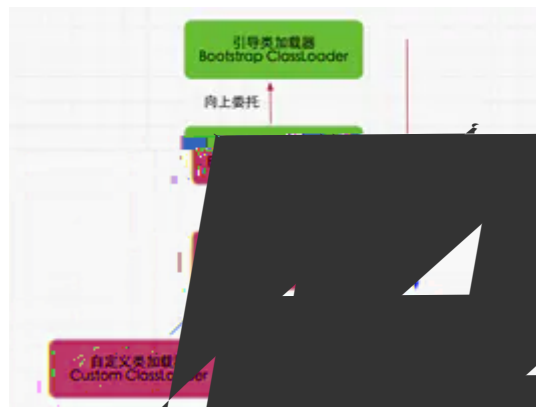
46. 下

- 为 下5个 :

- o : c.a ;
- o : c.a ;
- o : 中 ;
- o : 中 中 为 个
- o : 中 中

47. 什么 双亲 ?

- 之 下 于 个 , 个 个
JVM 中 个 JVM , 为 c.a



- :
 - o (B L , 中 且
 - o : -Xb .a
 - o : \.ib\e Ja a.e .di
 - o (E e i L : \.ib\e Ja a.e .di
 - o (A .ica i C.a L (c.a 上
- 个 : 个 了 , 不 个 ,
个 中 , (中) ,
- : 当一个类收到了类加 一个类, 而是将其委派给父类, 由父类去加 , 如果此时父类不能加 , 反 给 完成类的加 。

JVM 优

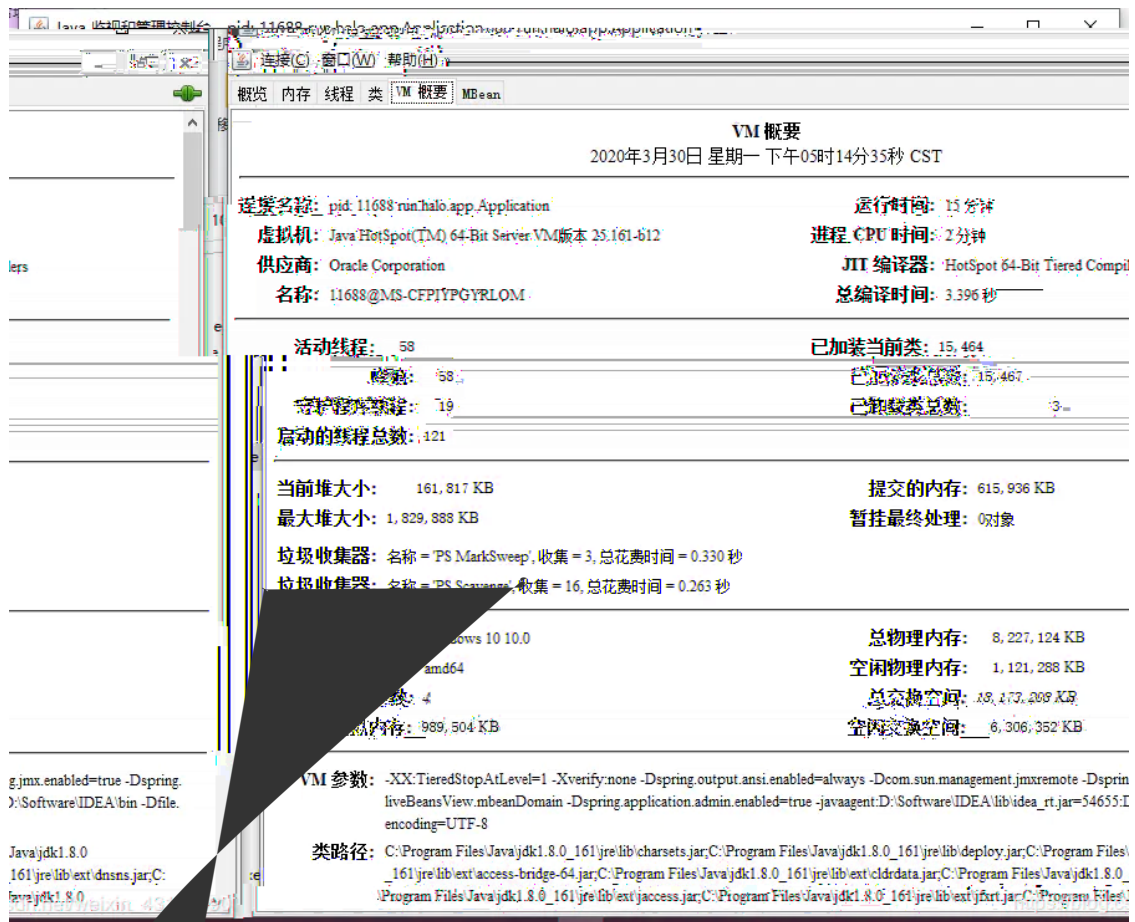
48. JVM 优 参 可以 参 值

- IDEA, Ec.i e,

```
java -xms1024m -xmx1024m ...等等等 JVM参数 -jar springboot_app.jar &
```

49. 下JVM 具?

- JDK 了 , 于JDK bi 下, 中 jc i d
- jc 于 JVM 中 ;



- o j i a JDK , :
- gc

! [在 插入图片描](<https://user-gold-cdn.xitu.io/2020/4/13/171729fd4d2a3018?imageView2/0/w/1280/h/960/format/webp/ignore-error/1>)

50. JVM 优 参 哪些?

#常用的设置

-Xms: 初始堆大小, JVM 启动的时候, 给定堆空 大小。

-Xmx: 最大堆大小, JVM 运行过程中, 如果初始堆空间不够的时候, 最大可以扩展到多少。

-Xmn: 设置堆中年 代大小。整个堆大小=年 代大小+年老代大小+持久代大小。

-XX:NewSize=n 设置年轻代初始化大小

-XX:MaxNewSize=n 设置年轻代最大值

-XX:NewRatio=n 设置年轻代和年老代的比值。如：-XX:NewRatio=3，表示年轻代与年老代比值为 1:3，年轻代占整个年轻代+年老代和的 1/4

-XX:SurvivorRatio=n 年轻代中 Eden 区与两个 Survivor 区的比值。注意 Survivor 区有两个。8 表示两个Survivor :eden=2:8 ,即一个Survivor占年轻代的1/10，默认就为8

-Xss: 设置每个线程的堆栈大小。JDK5后每个线程 Java 栈大小为 1M，以前每个线程堆栈大小为 256K。

-XX:ThreadStackSize=n 线程堆栈大小

-XX:PermSize=n 设置持久代初始值

-XX:MaxPermSize=n 设置持久代大小

-XX:MaxTenuringThreshold=n 设置年轻代垃圾对象最大年龄。如果设置为 0 的，则年轻代对象不经 Survivor 区，直接进入年老代。

#下 是一些不常用的

-XX:LargePageSizeInBytes=n 设置堆内存的内存大小

-XX:+UseFastAccessorMethods 优化原始类型的getter方法性能

-XX:+DisableExplicitGC 禁止在行期显式地使用System.gc()，默认启用

-XX:+AggressiveOpts 是否启用JVM开发团队最新的优化成果。例如编译优化，偏向，并行年老代收等，jdk6之后默认启动

-XX:+UseBiasedLocking 是否启用偏向，JDK6 默认启用

-Xnoclassgc 是否禁用垃圾回收

-XX:+UseThreadPriorities 使用本地线程的优先级，默认启用

等等等.....

51. JVM GC

- - :+U e C
- o

-XX:+UseSerialGC: 设置串行收 器，年 带收 器

-XX:+UseParNewGC: 设置年 代为并行收 。可与 CMS 收 同时使用。JDK5.0 以上，JVM 会根据系统置自行设置，所以无 再设置此值。

-XX:+UseParallelGC: 设置并行收 器，目标是目标是 到可控制的吞吐

-XX:+UseParallelOldGC: 设置并行年老代收 器，JDK6.0 支持对年老代并行收 。

-XX:+UseConcMarkSweepGC: 设置年老代并发收 器

-XX:+UseG1GC: 设置 G1 收 器，JDK1.9 认垃圾收 器