类加载

2019年6月1日 星期六 18:59

任何程序都需要加载到内存中才能和CPU进行交流。

字节码主要指令

- 加载或存储指令
- 运算指令
- 类型转换指令
- 对象创建和访问指令
  - 创建对象
  - 访问属性
  - 检查实例数据
- 操作栈管理指令
- 方法调用与返回
- 同步指令

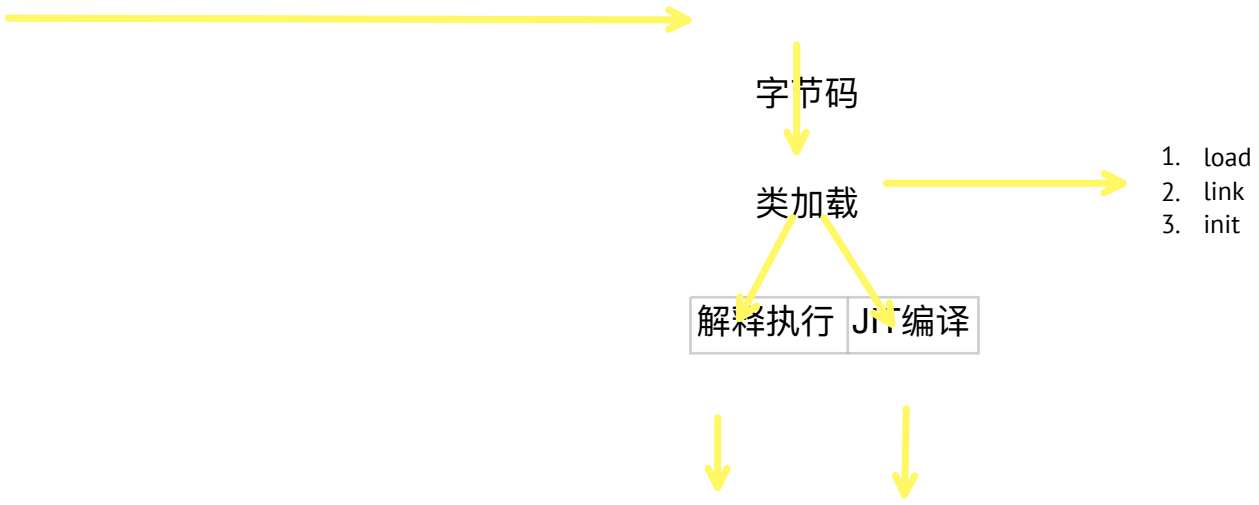
Java源文件

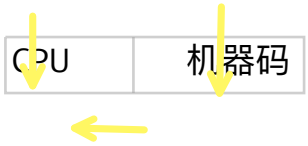
词法解析：通过空格分隔出单词，操作符，控制符等信息

语义分析：检查关键字是否合理，类型是否匹配，作用域是否正确

生成字节码

加载到JVM环境上执行





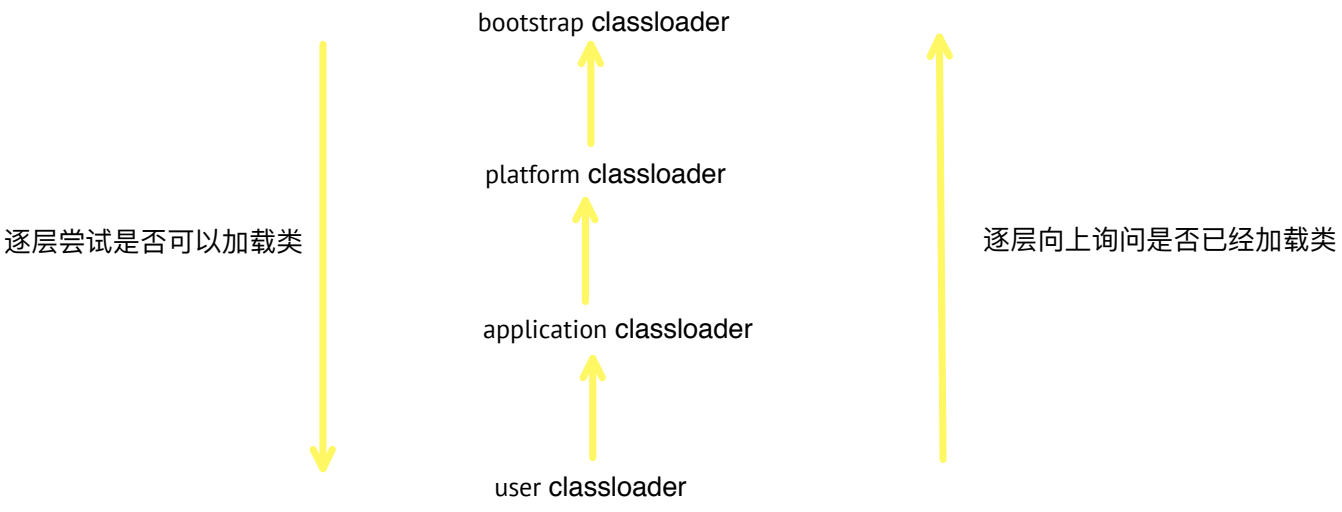
类加载过程

字节码需要通过classloader加载到内存之后才可以实例化。
类加载是将字节码实例化成Class对象并进行初始化的过程。

- load：读取类文件，产生二进制流，初步校验，创建实例
- link
  - 验证：更详细的校验
  - 准备：为静态变量分配内存，并设定默认值
  - 解析：解析类和方法，确保类与类之间引用正确，完成内存结构布局
- init：执行类的<clinit>方法

类加载器

- bootstrap
  - 负责装载最核心的Java类，如object、string、system等
- platform classloader
  - 平台类加载器，用以加载一些扩展的系统类
- application classloader
  - 引用类加载器，加载用户定义在classpath上的类



双亲委派模型

自定类类加载器的作用

- 隔离加载类
- 修改类加载方式
- 扩展加载源
- 防止源码泄漏

网络

2019年6月1日 星期六 18:48

应用层	打包数据

↓

传输层	加上端口号

↓

网络层	加上ip地址

↓

数据链路层	加上mac地址

↓

物理层

↓

三次握手

确保信息对等，防止超时

A机器 B机器

请求连接 SYN=1,seq=x →

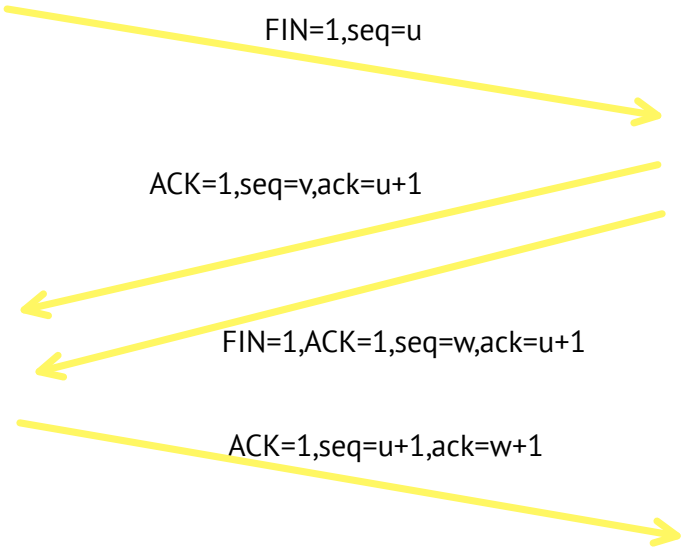
SYN=1,ACK=1,seq=y,ack=x+1 ←

ACK=1,seq=x+1,ack=y+1 →

四次挥手

A机器　　B机器

请求关闭



垃圾回收

2019年6月1日 星期六　17:33

对象是否存活
- 引用计数
- 可达性分析

引用
- 强引用
- 软引用
- 弱引用
- 虚引用

任何对象的finalize()方法只会被执行一次

方法区垃圾回收
- 废弃常量
- 无用的类
  - 该类的所有实例都被回收，堆中没有该类的实例
  - 加载该类的classloader已经被回收
  - 该类的所有对象都没有被引用，无法在任何地方通过反射访问该类

垃圾回收算法

标记-清除

缺点：效率低，回收后会产生不连续的内存碎片，遇到较大对象之后会再一次出发gc

对象没有与gc roots 的引用链

第一次标记，判断是否需要执行finalize

否（对象没有覆盖finalize或已经被调用过）

是（覆盖了finalize并且没有被调用过）

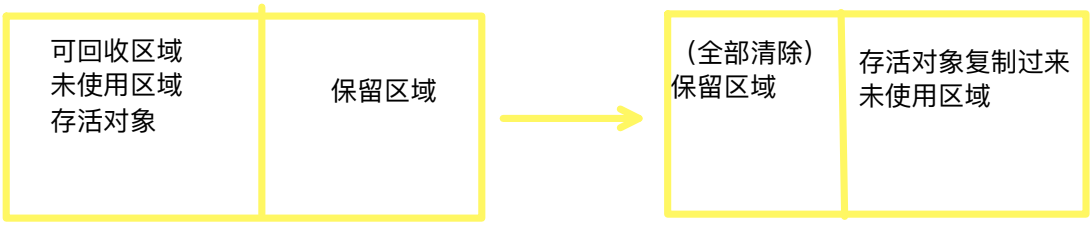
放在f-queue中，由finalize线程执行finalize方法

gc对引用链上的任何对象建立连接，移除即将回收的对象

2. 复制算法

缺点：每次将一半区域作为保留区域代价太大

现今虚拟机一般用这种方法回收新生代，以8:1:1分为Eden和两个survivor区域，每次使用一个Eden和一个survivor，回收时将所有存活对象复制到另一个survivor区域，如果另一个survivor区域空间不足，可以根据分配担保进入老年带

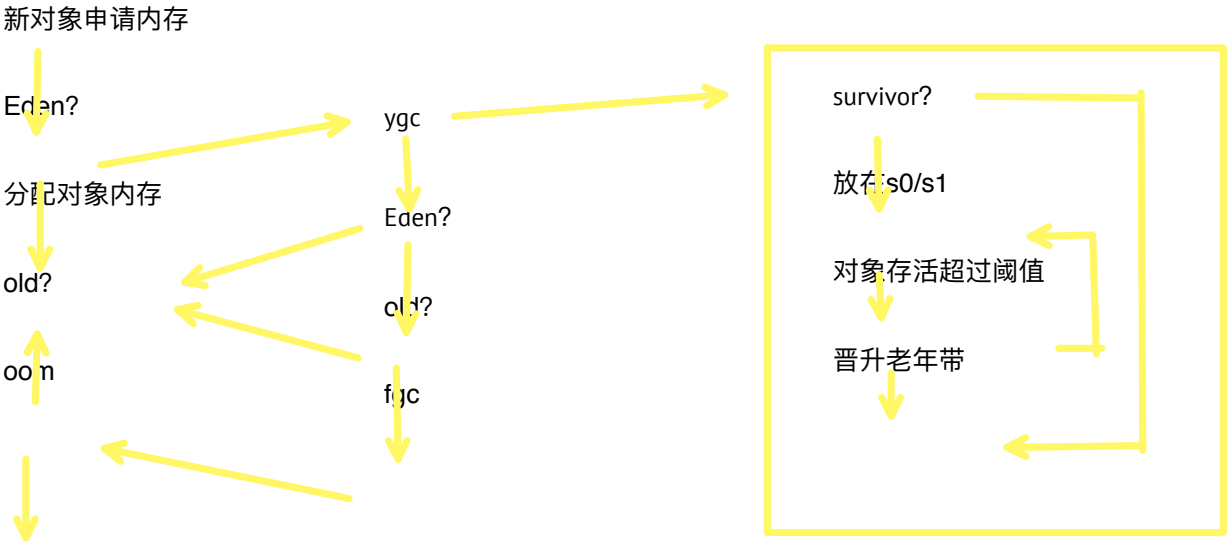


3. 标记-整理

前面仍然进行标记过程，但后续不进行直接清理，而是让所有存活的对象向一段移动，然后再进行清理

分代收集

一般根据对象存活周期不同分为新生代和老年带，新生代由于存活对象少，可以用复制算法，老年可以采用标记-整理



常见的垃圾回收器

Sreial

主要用于YGC，采用转行单线程。垃圾回收的某个过程会暂停整个应用程序的执行，即“stop the world”。

CMS（concurrent mark sweep collector）

回收停顿时间较短，目前较常用。通过初始标记，并发标记，重新标记，并发清除四个步骤完成回收。

G1

Hotspot在JDK7中推出新一代G1垃圾回收器。和CMS相比，G1具备压缩功能，能避免碎片问题，暂停时间更加可控。

内存布局

2019年6月1日 星期六 17:01

堆

线程共享

- 新生代
  - Eden
  - S0
  - S1
- 老年代

方法区

方法区的内存回收主要针对常量池的回收和类的卸载

- 运行时常量池
- 类信息
- 常量
- 静态变量
- 即时编译器编译后的代码

虚拟机栈

线程私有，由栈帧组成，一个方法一个栈帧

- 局部变量表
  - 方法参数
  - 局部变量
- 操作栈
  - 执行字节码指令
- 动态链接
- 出口

程序计数器

线程私有

对象的创建

2019年6月1日 星期六 17:11

new

检查指令的参数是否能在常量池中找到一个类的符号引用

检查类是否已经被加载

分配内存

堆内存规整：指针碰撞

堆内存不规整：空闲列表

初始化位零值

对对象进行设置

设置对象头，包括类、类的元数据、hashcode、gc年代等

初始化

对象

对象头

对象自身运行时数据

hashcode、gc年代、线程锁、偏向线程id等

2. 实例数据
3. 对齐填充

对象访问定位