# 概述

# Hotspot虚拟机对象创建

## 对象创建过程

* 判断类是否加载，如果没有加载，先加载，否则执行下一步
* 申请内存，根据类信息在堆空间申请指定大小的内存空间；
* 内存空间清零；
* 调用类的init方法（注意init和clinit的区别）；
* 调用构造方法（可选）；

## 对象内存布局

一个对象在内存中的布局分为三个部分：

* 对象头
* 实例数据
* 对齐填充（非必须）

# 垃圾回收（Garbage Collection）

## 垃圾回收区域

* 堆
* 方法区

## 判断对象是否可回收

* 引用计数法，缺点是无法解决循环引用问题；
* 可达性分析法，从gc root节点向下搜索，没搜索一个节点，则标记该对象不能被回收。

Gc root包括如下几类（分别来自于方法区、虚拟机栈区、本地方法栈）：

* + 方法区Static变量引用的对象
  + 方法区的常量引用的对象
  + 虚拟机栈中本地栈帧中引用的对象
  + 本地方法栈中引用的对象；

## 引用

* 强引用：如果一对象被强引用，JVM垃圾回收器将不会回收该对象，即使是内存不足，JVM宁可抛出异常OutOfMemory,也不会回收该对象；
* 软引用：被软引用的对象，如果JVM内存不够使用，JVM将把被软引用的对象放入垃圾回收器中进行重新回收，如果再次回收之后内存仍然不够使用，将抛出OutOfMemory异常。如果GC回收时，内存足够使用，则被软引用的对象不会有影响；
* 弱引用：被弱引用的对象，GC回收器回收垃圾对象时，无论当前内存是否够用，被弱引用的对象都会被回收；
* 虚引用：

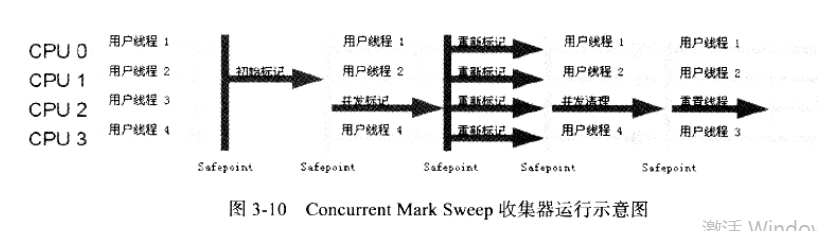
## 回收算法

* 标记-清除：首先找出未被引用的对象并标记，接下来把所有标记的对象从内存清除（导致内存碎片）；
* 复制算法：将内存平分为两块，一块内存使用完之后进行对象标记，接下来将未被标记的对象复制到另一块内存（内存使用率最大为50%）；
* 标记-整理：先进行标记，然后将存活的对象往一侧移动，
* 分代收集：将内存分为老年代和新生代，新生代采用复制算法，老年代采用标记-清除或者标记-整理算法

## 垃圾收集器

主要分新生代垃圾收集器，老年代垃圾收集器，没有万能的收集器，只有合适的收集器。重点掌握CMS（Concurrent Mark Sweep）和G1（Garbage First）收集器；

### CMS收集器



# 类加载

# 参考

《深入理解Java虚拟机：JVM高级特性与最佳实践》