## Java内存区域与内存溢出

1. 数据区划分
   1. 程序计数器
   2. 方法区（常量池）
   3. 堆
   4. 本地方法栈
   5. java虚拟机栈
2. 常见的内存溢出异常
   1. 堆溢出，OutOfMemoryError(Java heap space)
   2. 虚拟机栈和本地方法栈溢出：OutOfMemoryError()，StackOverflowError
   3. 方法区和运行时常量池溢出：OutOfMemoryError(PermGen space)

### 垃圾收集器与内存分配策略

1. 判断对象是否存活的算法
   1. 引用计数法
   2. 可达性分析法，GC roots（静态属性，常量，引用）
2. 对象的引用类型
   1. 强引用：new
   2. 软引用：有用但非必须，内存溢出异常前会清理
   3. 弱引用：有用性弱于软引用，在下次GC时清理
   4. 虚引用：此次回收
3. 堆内存分代
   1. 老年代
   2. 新生代
      1. Eden区，Survivor区
4. 垃圾收集算法
   1. 标记-清除
   2. 标记-复制
   3. 标记-整理
5. 几种常见的垃圾收集器以及常用的搭配
   1. Serial + Serial Old
   2. Serial + CMS
   3. ParNew + Serial Old
   4. ParNew + CMS
   5. Parallel Scavenge + Parallel Old
   6. Parallel Scavenge + Serial Old
   7. G1, ZGC, Shenandoah

#### 虚拟机性能监控、故障处理工具

1. 基础故障处理工具

##### 调优案例分析与实战

###### 类文件结构

1. 虚拟机类加载机制
2. 类的加载过程
   1. 加载
      1. 通过一个类的全限定名去获取定义此类的二进制字节流
      2. 将字节流中的静态存储结构转化为方法区的运行时数据结构
      3. 在内存中生成一个代表这个类的java.lang.Class对象，作为方法区这个类的数据访问入口
   2. 验证
      1. 文件格式验证（开头，版本号，常量类型）
      2. 元数据验证（继承关系校验）
      3. 字节码验证
      4. 符号引用验证
   3. 准备
      1. 赋予静态变量以默认初始值，如果变量被final修饰，将赋予定义的值
   4. 解析（符号引用，直接引用）
      1. 类或接口的解析
      2. 字段解析
      3. 方法解析
      4. 接口方法解析
   5. 初始化
   6. 执行
   7. 卸载
3. **虚拟机字节码执行引擎**
4. **类加载及执行子系统的案例与实战**