- JVM
  - 内存管理
    - 区域
      - 方法区（共享）
        - 存储已经被加载的类信息、常量、静态变量、编译后的代码缓存、方法描述
          - 子主题 1
        - 运行时的常量池
          - **运行时，常量池无法申请到内存时 OutOfMemoryError**
      - 堆
        - 新生代、老年代、永久、Eden、Survivor
      - 虚拟机栈：
        - 存放每个执行java方法的内存信息，为虚拟机执行java方法服务
          - 操作数栈
          - **局部变量表**
            - Java虚拟机基本数据类型
            - 对象引用
            - returnAddress
          - 动态链接、方法进出口
      - 本地方法栈
        - 为虚拟机使用到的本地方法服务
      - PC
        - 存放编译期间生成的各种字面量与符号引用
    - 溢出异常
      - 堆溢出
        - 1. 分析堆转储快照，判断内存泄露or内存溢出
          - memory leak：内存泄露
            - 通过工具找到泄露对象到GC Root的引用链，从而定位代码
          - memory overflow: 内存溢出
            - 检查Java虚拟机的堆参数-Xms，-Xmx设置是否可以调整。
      - 虚拟机栈和本地方法栈溢出
        - stackOverflowError
          - 线程请求的栈的深度大于虚拟机允许最大深度
        - OutOfMemoryError
          - 如果虚拟机的栈内存允许动态扩展才可能发生
      - 方法区（运行时常量池 OutOfMemory
        - 常量池都是分配在永久代中，因此有一个最大限制
  - GC， 内存分配
    - 判断是否需要被回收
      - 对象是否已死（堆回收）
        - 引用计数（引用计数器），可达分析（GCRoots)
      - 类是否不再使用（方法区回收）
        - 该类所有实例已被回收、加载器被回收、这个类对应的java.langClass对象没有任何地方被引用（无法在任何地方通过反射访问该类的方法）
    - 引用关系
      - 强引用
        - 最普遍的，Object obj = new Object()
          - 只要引用关系在，GC就不回收
      - 软引用
        - 有用且非必需的对象
          - 系统内存溢出异常钱会第二次回收，若还没有足够内存才会异常
      - 弱引用
        - 比软引用强度弱，非必需
          - 只能生活在下一次垃圾收集发生为止
      - 虚引用
        - 不影响对象本身的生存时间，无法通过这个来获得对对象实例。
          - 为了在这个对象在GC回收时收到一个【系统通知】
    - 垃圾收集算法
      - 【标记清除算法】
        - 执行效率不稳定，随着对象数量增长而降低；碎片
      - 【标记复制】
        - 内存按容量分厂相等两块。缩小内存
      - 【标记整理】
        - 所有存活对象都向内存空间一端移动，清除边界以外的内存；会stop the world
    - 收集器
      - Serial
        - 垃圾回收时暂停所有工作
      - ParNew
        - 多线程并行版本的Serial
      - Parallel Scavenge
      - CMS
        - 获得最短回收停顿时间为目标的收集器
      - G1
        - 基于Region, region中又分成了eden, suvivor, old。
          - G1引入了region概念，每个region都是连续的虚拟内存范围。
    - 内存分配
      - 对象优先在新生代Eden区分配
        - 对象产生在Eden区，经过第一次Minor GC后依然存活且内能够被Survivor容纳，则会移动到Survivor空间，将年龄设置为1岁。每经过一次GC，年龄+1，>15时晋升到老年代
      - 老年代
        - 大对象、长期存活对象
  - 虚拟机性能监控
  - 加载机制
    - 类生命周期
      - 加载，验证，解析，准备，初始化，使用， 卸载
    - 类加载
      - 加载
        - 生成一个java.langClass对象作为方法区这个类的各种数据的访问入口
      - 验证
        - class文件字节流中信息符合约束要求，防止有恶意企图的字节码
      - 准备
        - 为类中变量分配内存（static变量），初始化一些系统要求的零值
      - 解析
        - jvm将常量池中符号引用替换为直接引用
      - 初始化
        - 根据代码初始化类变量和其他资源
    - 类加载器
      - 双亲委派
  - 并发
    - volatile
      - 原子 可见 有序
    - 锁