《深入理解Java虚拟机》学习总结

- **-classic**
  - `-classic` 参数其实是选择使用 `classic VM` 虚拟机的意思；但是 `classic VM` 已经从jdk1.4中移除；

- **关于Android VM**
  - `Android` 使用的是 `Dalvik VM`
  - `Dalvik` 与 `JVM` 的区别在于
    - Dalvik基于 `寄存器` ：选择基于寄存器的方式是因为它对*提前优化*（*ahead-of-time optimization*）提供了更好的支持，而这对类似于移动电话这样的受限环境是颇有裨益的；
      - 寄存器是CPU上面的一块存储空间；
    - JVM基于 `栈` ：栈是内存上面的一段连续的存储空间；
    - JVM相关执行文件类型： `.java` 文件 -> `.class` 文件 -> `.jar` 文件；
    - Dalvik相关执行文件类型： `.java` 文件 –> `.class` 文件 -> `.dex` 文件；

- **JRockit**
  - `JRockit` 由 `BEA` 公司开发， `JRockit` 专注于服务器领域，所以不在乎虚拟机的启动速度，所以 `JRockit` 内部不包含 `解析器` 的实现，全部代码靠 `即时编译器` 编译后执行；
  - `JRockit` 的 `垃圾回收器` 和 `MissionControl` 服务组件的实现，在众多JVM中出于领先水平；

- **IBM J9 VM**
  - `IBM J9 VM` 是内部代号，正式名称是 `IT4J` （IBM Technology for Java Virtual Machine）；
  - `IBM J9 VM` 和Hotspot类似，考虑场景全面（服务器、桌面、移动设备等）；

- **Zing VM**
  - Zing VM是在Hotspot基础上进行了大量的优化改进，是真正的“高性能”

- **Liquid VM**
- **运行在JVM上的语言**
  - JESS、Lisp、groovy、icon、E、Tiger、Tcl、Eiffel、v-language、jRuby、PHP、LLP、Jython、Piccola、Phobos、ObjectScript、Yoix、G、WebL、JavaFX、BeanShell、Pascal、Scala、FScript、Nice、Zigzag、Jickle、JudoScript、Oberon、Sather、Sleep、Bex Script、Pnuts、Anvil、Dawn、Forth、C#、Yassl、JHCR、CAL、Simkin；
  - 同一虚拟机上 `非Java语言` 与Java之间的交互是比较容易的，但 `非Java语言` 之间的交互就比较烦琐， `dynalang` 项目的出现就是为了解决这个问题；

- **JDK 64bit**
  - 由于 `指针膨胀` 和各种 `数据类型对齐补白` 等原因，运行在 `64bit系统` 上的Java应用会额外增加10%~30%内存消耗，
  - `64bit虚拟机` 几乎全面落后 `32bit虚拟机` 15%性能；
  - `sun` 公司也注意到了这个问题，提供了一个命令参数 `-XX:+ UseCompressedOops` （普通对象指针压缩功能，目的是为了降低 `64bit` 内存消耗）；
    - 但是，不建议显示设置，维持默认就好
    - 因为，此功能在执行代码时，动态植入压缩指令以节省内存消耗，但却增加了代码执行数量，所有Java堆内对象的指针都会被压缩，这些被压缩的指针需要更多的指令才能访问，而且并不只是读写字段才受影响，在实例方法调用、子类型检查等操作中也受影响（对象实例指向对象类型的指针也被压缩了）；

- **Open JDK衍生版本**
  - IceTea、UltraViolet