JVM

1. 数据类型

- 编译器在 `编译` 期间应当尽最大努力完成可能的 `类型检查`，使得虚拟机在运行期间无需再进行这些操作；
- `reference` 类型表示一个对象的 `引用`，可以想象成指向对象的 `指针`；
  - `reference` 和 `int`、`long`、`float`、`double` 等基本类型是一个层次的；前者是具体的数据类型，后者是某种数据类型的统称；（*书本P5*）
- jvm数据类型分为 `原始类型` 和 `引用类型`

  原始类型 包括：
  - 数值类型（ `numeric type` ）
    - 整型：
      - byte、short、int、long；（默认值为0；以有符号的二进制补码的形式存储）
      - char类型：16位无符号整形表示，即：2个字节表示UTF-16基本平面码点；默认值是null的码点（Unicode中null的码点，\u0000）
    - 浮点型：float、double；（默认值为正数0）
      - `float` 和 `double` 在内存中的存储形式采用了 `IEEE 754` 标准：*float*（ `符号位` + `8位幂值` + `23位小数位` ）、*double*（ `符号位` + `11位幂值` + `52位小数位` ）
      - float的幂值范围为 `-126` ~ `127`；8个幂值位全为 `0` 或全为 `1` 时表示特殊值；
      - `java` 中 `float` 必须使用 `f` 标注，否则表示 `double` 类型的计算；
  - `boolean` 类型：默认值为 `false`；
  - `returnAddress` 类型：指向某个 `操作码（opcode）` 的指针；此操作码与 `jvm的指令` 相对应；
- `float` 类型**存储结构**

  阅读这篇博客便可详细了解 `float` 类型存储结构，这里主要备注一下特殊情况：
  - 8位幂值全为0，并且小数部分是0，则表示 `± 0`（正负和符号位有关）
  - 8位幂值全为1，并且小数部分是0，则表示 `±无穷大`（正负和符号位有关）
  - 8位幂值全为1，并且小数部分非0，则表示 `NaN`
- 关于**浮点集合**和**扩展指数集合**

  表 2-1　浮点数集合的参数

参数	单精度浮点数集合	单精度扩展指数集合	双精度浮点数集合	双精度扩展指数集合
N	24	24	53	53
K	8	⩾ 11	11	⩾ 15
E_{max}	+127	⩾ +1 023	+1 023	⩾ +16 383
E_{min}	−126	⩽ −1 022	−1 022	⩽ −16 382

  上图中的几种集合是经过精心设计的，从左到右是被包含的关系：
  - `单精度浮点数集合` < `单精度扩展指数集合` < `双精度浮点集合` < `双精度扩展指数集合`
- `NaN` 与任何数进行比较和等值操作都会返回 `false`，包括 `NaN` 自己，eg：NaN == NaN – `false`
- jvm中没有提供任何boolean值专用的指令，`boolean` 编译后都是用 `int` 代替；
  - true → 1；　false → 0；
- jvm中可以创建boolean数组，通过公用 `byte数组` 的 `baload` 和 `bastore` 指令进行操作；
- **引用类型**

    - `类`：指向动态创建类实例
    - `数组`：指向数组实例
    - `接口`：指向实现了某接口的类或数组实例
- `引用类型`默认为`null`；*jvm规范并没有规定null在虚拟机中应该如何编码表示；*
- **数组类型**
    - 组件类型：int[][]的组件类型就是int[]，int[]的组件类型是int[]；
    - 元素类型：当组件不再是数组的时候，就是元素类型，如：int[]的元素类型是int；

2. 运行时数据区

有些数据区会随着jvm启动而创建，随着jvm退出而销毁；另外一些和线程一一对应，随`线程`的开始和结束而创建和销毁；

2.1 PC寄存器

- 每个线程一个PC寄存器

2.2 Java虚拟机栈

- 每个线程一个Java虚拟机栈（Java栈），和传统栈功能一样，用于保存局部变量和一些计算中间量；
- 除了栈帧出栈和入栈之外，虚拟机栈不会再受其他因素影响，所以可以栈帧可以在堆中分配；
- 虚拟机栈所使用的内存不必保证连续；
- Java栈可以设计为固定长度，也可以动态扩展和收缩；虚拟机的实现应该提供给使用者调节Java栈的手段；
- 创建栈相关异常
    - 如果是固定长度的栈，当请求分配栈容量超过虚拟机允许的最大容量，jvm抛出`StackOverflowError`；
    - 如果Java栈设计成了动态的，在尝试扩展的时候如果内存不足，则jvm抛出`OutOfMemoryError`；

3. 运行模式

jvm有2种运行模式：server和client 传送门1 传送门2

其他

- `class`文件中有一些`惯例`，比如：`字节序`的选用，这样做事为了统一某些操作，如此才能更好地做到`平台无关性`；

说明

- *文中的所有页码都是指`《java虚拟机规范 java se8》`中文版对应页码；*