/\*\*

\* JVM执行引擎

\*

\* 执行引擎是JVM中最核心的内容之一，描述了一个高层的抽象的方法调用及字节码执行的概念模型

\*

\* 这个概念模型是平台独立，与具体的硬件无关，甚至可以执行一些硬件不支持的指令集，主要是通过软件的形式实现

\*

\* 虚拟机规范中只是规定执行引擎的统一外观，但是并没有规定具体的实现方式 常见的执行引擎主要有以下三种实现方式： 1、解释型 2、即时编译型

\* 3、解释与即时编译相结合

\*

\* 运行时栈帧结构 栈帧是用于虚拟机支持方法调用和方法执行的数据结构，它是虚拟机线程栈栈中的一个元素

\*

\* 栈帧(Stack Frame)包含以下信息： 1、方法的局部变量表 2、操作数栈 3、动态连接 4、方法返回地址

\*

\*

\* 局部变量表 局部变量表，是用于存放方法参数及方法内定义的局部变量，他的容量在编译时就已经确定。

\* 方法表中Code属性的max-locals定义了最大容量,虚拟机定义的容量的基本单位是Slot,但是并没有规定Slot占用多大内存

\*

\* 虚拟机规范规定

\* boolean、byte、char、short、int、float、reference、returnAddress应该存放一个Slot中

\* long、double放在两个Slot中，这个不需要原子性，因为这是线程私有的，不会导致数据安全的问题

\*

\* Slot是可以重用的，这样不仅可以节省栈的空间，也可以影响垃圾回收的行为

\* 注意：方法内的定义变量并没有默认值，不过这在IDE中编译器一般会提示我们。

\*

\*

\* 操作数栈 操作数栈也称操作栈，是一个FIFO的栈，用于存储各种java数据类型的数据

\* 方法开始运行时这个栈是空的，栈中数据的放入及取出都是由方法中指令来完成的

\*

\* 动态连接

\*

\* 方法返回地址

\*

\* 附加信息 这部分规范中并没有定义，虚拟机实现者可以自己决定，如添加调试信息等

\*

\* JVM符号引用解析

\* 解析分为静态解析与动态解析

\* 调用目标方法唯一且确定时，会在类加载时进行解析，这种解析称为静态解析，包括类方法、私有方法、final方法，构造函数(<init>)、父类中的方法

\* 其他方法的符号引用的解析都是在运行时进行的，我们称之为动态解析

\*

\* 分派(Dispatch)

\* JVM在确定具体调用哪个方法的过程称为分派

\* 分派的类型：静态、动态、单分派、多分派

\*

\* 宗量

\* 方法的接收者和方法的参数，统称为宗量

\* 分派过程中依据一个宗量进行分派称之为单分派，依据多个宗量分派称之为多分派

\*

\* 重载：静态多分派

\* 重写：动态单分派（JDK1.6中式这种情况，JDK7中为了支持动态语言，添加了动态多分派）

\*

\* 动态解析具体方法：

\*

\* 稳定优化：使用了vtable,即虚表

\* 激进优化：内敛缓存与基于类型层次关系分析技术的守护内敛

\*

\* JVM字节码指令执行引擎

\* JVM执行引擎可分为：解释型与编译型两种引擎，具体的虚拟机实现可以自行选择，或是两者都有

\*

\* 指令集

\* 指令集可以分为基于栈的指令集与基于寄存器的指令集

\* 所有PC机的指令集都属于基于寄存器的指令集

\*

\* 基于栈的指令集可以运行在不同架构的机器上，比较通用，但是需要较多的指令及内存访问的开销（栈在内存中分配）

\* 基于寄存器的指令集指令少、高效，但是不通用

\*

\*/