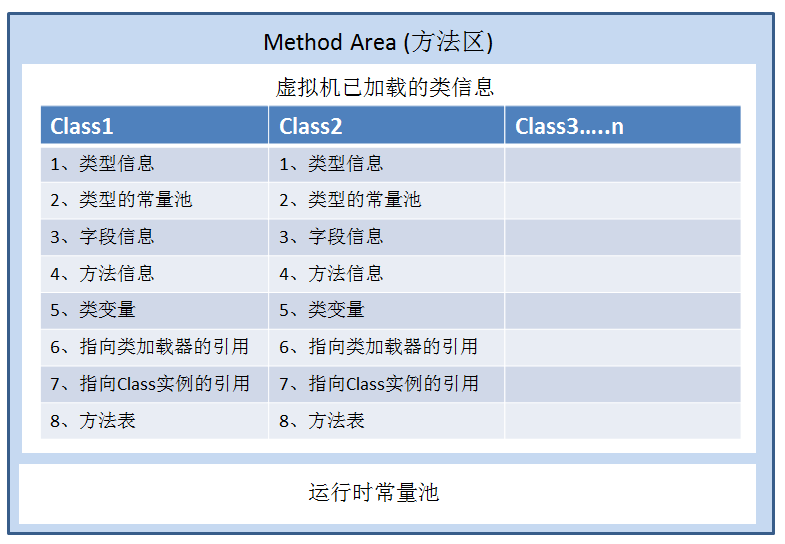
**JAVA虚拟机方法区介绍**

1、JVM内存模型

    大多数JVM将内存分配为Method Area(方法区)、Heap(堆)、Program Counter Register(程序计数器)、JAVA Method Stack(JAVA方法栈)、Native Method Stack(本地方法栈)。

1.1 Method Area  
    线程共享，存储已经被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等等。（HotSpot虚拟机上开发部署人员更愿意成为“**永久代”**，Permanent Generation）。示意图如下：  


1、类型信息

* 类型的全限定名
* 超类的全限定名
* 直接超接口的全限定名
* 类型标志（该类是类类型还是接口类型）
* 类的访问描述符（public、private、default、abstract、final、static）

2、类型的常量池**（该部分是独有的，然后运行时，把该部分加载进运行时常量池，当调用时则从符号引用解析为直接引用，但是有些确定的方法会直接转换，比如静态方法，比如构造方法）**

         存放该类型所用到的常量的有序集合，包括直接常量（如字符串、整数、浮点数的常量）和对其他类型、字段、方法的符号引用。常量池中每一个保存的常量都有一个索引，就像数组中的字段一样。因为常量池中保存中所有类型使用到的类型、字段、方法的符号引用，所以它也是**动态连接(栈中对应的方法指向这个引用)的主要对象（在动态链接中起到核心作用）。**

3、字段信息（该类声明的所有字段）

* 字段修饰符（public、protect、private、default）
* 字段的类型
* 字段名称

4、方法信息

         方法信息中包含类的所有方法，每个方法包含以下信息：

   • 方法名   
       • 方法的返回类型（包括void）   
       • 方法参数的类型、数目以及顺序   
       • 方法修饰符（public,private,protected,static,final,synchronized,native,abstract）  
针对非本地方法，还有些附加方法信息需要存储在方法区内：  
       • 方法字节码   
       • 方法中局部变量区的大小、方法栈帧   
       • 异常表

5、类变量（静态变量）

         指该类所有对象共享的变量，即使没有任何实例对象时，也可以访问的类变量。它们与类进行绑定。

6、  指向类加载器的引用

         每一个被JVM加载的类型，都保存这个类加载器的引用，类加载器动态链接时会用到。

7、指向Class实例的引用

        类加载的过程中，虚拟机会创建该类型的Class实例，方法区中必须保存对该对象的引用。通过Class.forName(String className)来查找获得该实例的引用，然后创建该类的对象。

8、方法表

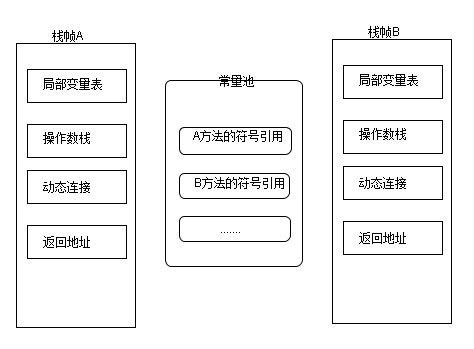
         为了提高访问效率，JVM可能会对每个装载的非抽象类，都创建一个数组，数组的每个元素是实例可能调用的方法的直接引用，包括父类中继承过来的方法。这个表在抽象类或者接口中是没有的，类似C++虚函数表vtbl。

二、运行时常量池(Runtime Constant Pool)

         Class文件中除了有类的版本、字段、方法、接口等描述信息外，还有一项信息是**运行时常量池**，用于存放**编译器生成的各种字面常量和符号引用**，这部分内容被类加载后进入方法区的运行时常量池中存放。

运行时常量池相对于Class文件常量池的另外一个特征具有动态性，可以在运行期间将新的常量放入池中（典型的如String类的intern()方法）。

运行时常量池是把Class文件常量池加载进来，每个类有一个独立的。刚开始时运行的时候运行时常量池里的链接都是符号链接（只用名字没有实体），跟在Class文件里的一样；只有调用该方法时，才把常量转换成直接引用。然后就可以供给给其他方法调用了。如下3个图：



如果符号引用是在**类加载阶段或者第一次使用的时候转化为直接应用**,那么这种转换成为**静态解析**,如果是在**运行期间转换为直接引用**,那么这种转换就成为**动态连接。**

例如说加载完Class A之后的Foo.bar()方法等一段时间被调用，A.class文件里就会有对该方法的Method ref常量，是个符号链接（只有名字没有实体），加载到运行时常量池也还是一样是符号链接（符号引用），等真的要调用该方法的时候该常量就会被resolve（解析）为一个直接链接（直接引用））。

**（该部分是独有的，然后运行时，把该部分加载进运行时常量池，当调用时则从符号引用解析为直接引用，但是有些确定的方法会直接转换，比如静态方法，比如构造方法）**