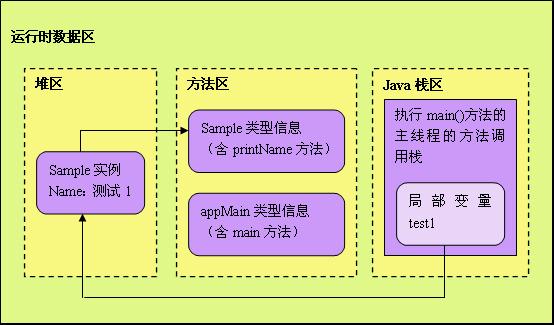
**[java虚拟机内存的堆区（heap），栈区（stack）和静态区（static/method）](http://www.blogjava.net/yaoyaojj/archive/2011/07/31/355438.html)**

JAVA的JVM的内存可分为3个区：堆(heap)、栈(stack)和方法区(method)  
  
堆区:  
1.存储的全部是对象，每个对象都包含一个与之对应的class的信息。(class的目的是得到操作指令)  
2.jvm只有一个堆区(heap)被所有线程共享，堆中不存放基本类型和对象引用，只存放对象本身.  
  
3.一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。  
  
栈区:  
1.每个线程包含一个栈区，栈中只保存基础数据类型的对象和自定义对象的引用(不是对象)，对象都存放在堆区中  
2.每个栈中的数据(原始类型和对象引用)都是私有的，其他栈不能访问。  
3.栈分为3个部分：基本类型变量区、执行环境上下文、操作指令区(存放操作指令)。  
  
  
4.由编译器自动分配释放 ，存放函数的参数值，局部变量的值等．  
  
   
  
   
  
   
  
静态区/方法区:  
1.方法区又叫静态区，跟堆一样，被所有的线程共享。方法区包含所有的class和static变量。  
2.方法区中包含的都是在整个程序中永远唯一的元素，如class，static变量。  
  
3.—，全局变量和静态变量的存储是放在一块的，初始化的全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。  
  
  
代码实例（转自网络）  
  
AppMain.java  
 public   class  AppMain //运行时, jvm 把appmain的信息都放入方法区  
{  
    public   static   void  main(String[] args)  //main 方法本身放入方法区。  
    {  
         Sample test1 = new  Sample( " 测试1 " );   //test1是引用，所以放到栈区里， Sample是自定义对象应该放到堆里面  
          Sample test2 = new  Sample( " 测试2 " );  
  
         test1.printName();  
         test2.printName();  
    }  
}  
  
Sample.java  
  
public   class  Sample //运行时, jvm 把appmain的信息都放入方法区  
{  
     /\*\* 范例名称 \*/  
     private  name; //new Sample实例后， name 引用放入栈区里，  name 对象放入堆里  
  
      /\*\* 构造方法 \*/  
     public  Sample(String name)  
     {  
          this .name = name;  
     }  
  
     /\*\* 输出 \*/  
     public   void  printName()   //print方法本身放入 方法区里。  
     {  
         System.out.println(name);  
     }  
}  
  
代码的执行过程：

系统收到了我们发出的指令，启动了一个Java虚拟机进程，这个进程首先从classpath中找到AppMain.class文件，读取这个文件中的二进制数据，然后把Appmain类的类信息存放到运行时数据区的方法区中。这一过程称为AppMain类的加载过程。  
接着，Java虚拟机定位到方法区中AppMain类的Main()方法的字节码，开始执行它的指令。这个main()方法的第一条语句就是：  
Sample test1=new Sample("测试1");  
  
  
语句很简单啦，就是让java虚拟机创建一个Sample实例，并且呢，使引用变量test1引用这个实例。貌似小case一桩哦，就让我们来跟踪一下Java虚拟机，看看它究竟是怎么来执行这个任务的：  
  
  
1、 Java虚拟机一看，不就是建立一个Sample实例吗，简单，于是就直奔方法区而去，先找到Sample类的类型信息再说。结果呢，嘿嘿，没找到@@，这会儿的方法区里还没有Sample类呢。可Java虚拟机也不是一根筋的笨蛋，于是，它发扬“自己动手，丰衣足食”的作风，立马加载了Sample类，把Sample类的类型信息存放在方法区里。  
  
  
2、 好啦，资料找到了，下面就开始干活啦。Java虚拟机做的第一件事情就是在堆区中为一个新的Sample实例分配内存, 这个Sample实例持有着指向方法区的Sample类的类型信息的引用。这里所说的引用，实际上指的是Sample类的类型信息在方法区中的内存地址，其实，就是有点类似于C语言里的指针啦~~，而这个地址呢，就存放了在Sample实例的数据区里。  
  
   
   
  
   
3、 在JAVA虚拟机进程中，每个线程都会拥有一个方法调用栈，用来跟踪线程运行中一系列的方法调用过程，栈中的每一个元素就被称为栈帧，每当线程调用一个方法的时候就会向方法栈压入一个新帧。这里的帧用来存储方法的参数、局部变量和运算过程中的临时数据。OK，原理讲完了，就让我们来继续我们的跟踪行动！位于“=”前的Test1是一个在main()方法中定义的变量，可见，它是一个局部变量，因此，它被会添加到了执行main()方法的主线程的JAVA方法调用栈中。而“=”将把这个test1变量指向堆区中的Sample实例，也就是说，它持有指向Sample实例的引用。  
OK，到这里为止呢，JAVA虚拟机就完成了这个简单语句的执行任务。参考我们的行动向导图，我们终于初步摸清了JAVA虚拟机的一点点底细了，COOL！  
接下来，JAVA虚拟机将继续执行后续指令，在堆区里继续创建另一个Sample实例，然后依次执行它们的printName()方法。当JAVA虚拟机执行test1.printName()方法时，JAVA虚拟机根据局部变量test1持有的引用，定位到堆区中的Sample实例，再根据Sample实例持有的引用，定位到方法去中Sample类的类型信息，从而获得printName()方法的字节码，接着执行printName()方法包含的指令。

[**Java中六种数据存储方式**](http://www.cnblogs.com/leokou/articles/2070524.html)

存储数据  
1．寄存器（register）。这是最快的存储区，因为它位于不同于其他存储区的地方——处理器内部。但是寄存器的数量极其有限，所以寄存器由编译器根据需求进行分配。你不能直接控制，也不能在程序中感觉到寄存器存在的任何迹象。  
2． 堆栈（stack）。位于通用 RAM（random-access memory，随机访问存储器）中，但通过它的“堆栈指针”可以从处理器那里获得直接支持。堆栈指针若向下移动，则分配新的内存；若向上移动，则释放那些内存。这是一种快速有效的分配存储方法，仅次于寄存器。创建程序时，Java 编译器必须知道存储在堆栈内所有数据的确切大小和生命周期，因为它必须生成相应的代码，以便上下移动堆栈指针。这一约束限制了程序的灵活性，所以虽然某些 Java 数据存储于堆栈中——特别是对象引用，但是 Java 对象并不存储于其中。  
3． 堆（heap）。一种通用性的内存池（也存在于 RAM 区），用于存放所有的 Java 对象。堆不同于堆栈的好处是：编译器不需要知道要从堆里分配多少存储区域，也不必知道存储的数据在堆里存活多长时间。因此，在堆里分配存储有很大的灵活性。当你需要创建一个对象时，只需用 new 写一行简单的代码，当执行这行代码时，会自动在堆里进行存储分配。当然，为这种灵活性必须要付出相应的代价。用堆进行存储分配比用堆栈进行存储存储需要更多的时间（如果确实可以在 Java 中像在 C++中一样用栈保存对象）。  
4． 静态存储（static storage）。这里的“静态”是指“在固定的位置”（尽管也在 RAM 里）。静态存储里存放程序运行时一直存在的数据。你可用关键字 Static 来标识一个对象的特定元素是静态的，但 Java 对象本身从来不会存放在静态存储空间里。  
5． 常量存储（constant storage）。常量值通常直接存放在程序代码内部，这样做是安全的，因为它们永远不会被改变。有时，在嵌入式系统中，常量本身会和其它部分隔离开，所以在这种情况下，可以选择将其存放在 ROM（read-only memory，只读存储器）中。  
6． 非 RAM 存储（non-RAM storage）。如果数据完全存活于程序之外，那么它可以不受程序的任何控制，在程序没有运行时也可以存在。其中两个基本的例子是“流对象（streamed object）”和“持久化对（persistent object）”。在“流对象”中，对象转化成字节流，通常被发送给另一台机器。在“持久化对象”中，对象被存放于磁盘上，因此，即使程序终止，它们仍可以保持自己的状态。这种存储方式的技巧在于：把对象转化成可以存放在其它媒介上的事物，在需要时，可恢复成常规的、基于 RAM 的对象。

[复制代码](javascript:void(0);)

1、Register（寄存器）   
 存在与CPU中，存取速度最快，但数量有限   
  
  
2、Stack(栈、堆栈)   
 存在与内存中，存取的速度和效率仅次于Register，在程序加载的时候编译器必须准确知道所需的所有空间大小，通过指针来操作

3、Heap(堆中)   
 存在与内存中，存取方式灵活，无需编译器关心，速度比不上Stack   
  
  
4、Static (静态类型)   
 整个程序都可以访问，对象的属性可以是静态的，但JAVA对象不会是静态的.   
  
  
5、Constant（常量）   
 存在于程序代码中，不能改变   
  
  
6、非易失性数据   
 a、流类型数据   
 b 、持久性数据