Research stack and heap in java

jli

<java中堆栈的区别>

首先我们要明确，我们讲的不是java中LIFO的栈数据结构，而是jvm（虚拟）内存中的堆和栈。

1.功能不同

栈内存使用来存放局部变量和方法调用

对内存用来存储java中的对象。成员变量，局部变量，类变量，他们指向的对象都存储在堆内存中

2.独特有还是共享

栈内存归属于单个线程，每个线程都会有一个栈内存，其存储的变量只能在其所属线程中可见，即栈内存可以理解成线程的私有内存。  
而堆内存中的对象对所有线程可见。堆内存中的对象可以被所有线程访问。

3.异常错误类型不一样

4.空间大小

栈的内存要远远小于堆内存，如果你使用递归的话，那么你的栈很快就会充满。如果递归没有及时跳出，很可能发生StackOverFlowError问题。  
你可以通过-Xss选项设置栈内存的大小。-Xms选项可以设置堆的开始时的大小，-Xmx选项可以设置堆的最大值。

<java内存分配>

简单通俗的讲，一个完整的Java程序运行过程会涉及以下内存区域：

l 寄存器：JVM内部虚拟寄存器，存取速度非常快，程序不可控制。

l 栈：保存局部变量的值，包括：

（1）.用来保存基本数据类型的值；

（2）.保存类的实例，即堆区对象的引用(指针)。也可以用来保存加载方法时的帧。

l 堆：用来存放动态产生的数据，比如new出来的对象。注意创建出来的对象只包含属于各自的成员变量，并不包括成员方法。因为同一个类的对象拥有各自的成员变量，存储在各自的堆中，但是他们共享该类的方法，并不是每创建一个对象就把成员方法复制一次。

l 常量池：JVM为每个已加载的类型维护一个常量池，常量池就是这个类型用到的常量的一个有序集合。包括直接常量(基本类型，String)和对其他类型、方法、字段的符号引用(1)。池中的数据和数组一样通过索引访问。由于常量池包含了一个类型所有的对其他类型、方法、字段的符号引用，所以常量池在Java的动态链接中起了核心作用。常量池存在于堆中。

l 代码段：用来存放从硬盘上读取的源程序代码。

l 数据段：用来存放static定义的静态成员。

下面是内存表示图：



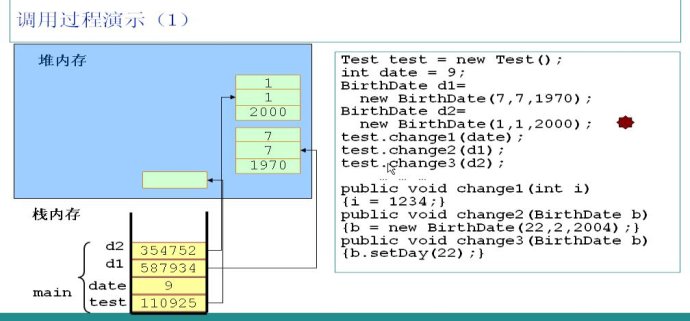
上图中大致描述了Java内存分配，接下来通过实例详细讲解Java程序是如何在内存中运行的。

预备知识：

1.一个Java文件，只要有main入口方法，我们就认为这是一个Java程序，可以单独编译运行。

2.无论是普通类型的变量还是引用类型的变量(俗称实例)，都可以作为局部变量，他们都可以出现在栈中。只不过普通类型的变量在栈中直接保存它所对应的值，而引用类型的变量保存的是一个指向堆区的指针，通过这个指针，就可以找到这个实例在堆区对应的对象。因此，普通类型变量只在栈区占用一块内存，而引用类型变量要在栈区和堆区各占一块内存。

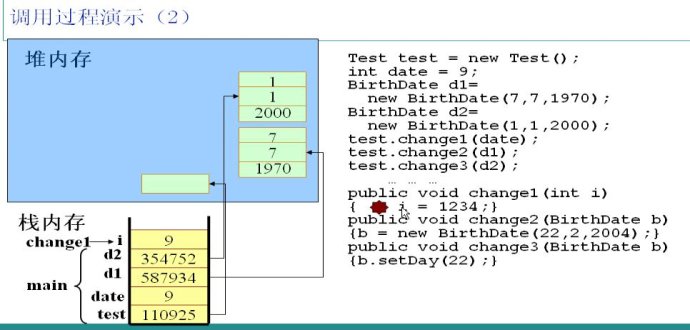
示例：



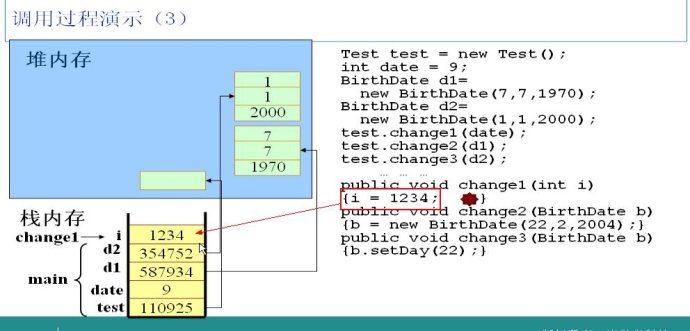
1.JVM自动寻找main方法，执行第一句代码，创建一个Test类的实例，在栈中分配一块内存，存放一个指向堆区对象的指针110925。

2.创建一个int型的变量date，由于是基本类型，直接在栈中存放date对应的值9。

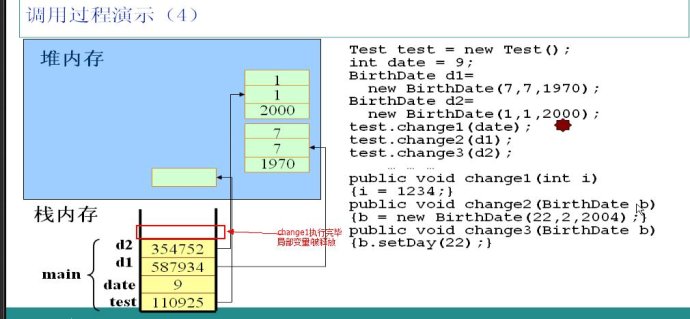
3.创建两个BirthDate类的实例d1、d2，在栈中分别存放了对应的指针指向各自的对象。他们在实例化时调用了有参数的构造方法，因此对象中有自定义初始值。



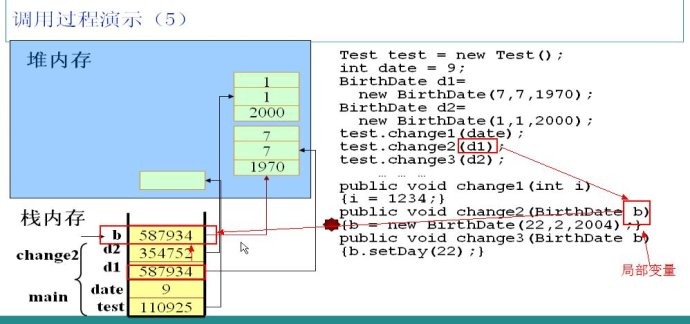
调用test对象的change1方法，并且以date为参数。JVM读到这段代码时，检测到i是局部变量，因此会把i放在栈中，并且把date的值赋给i。



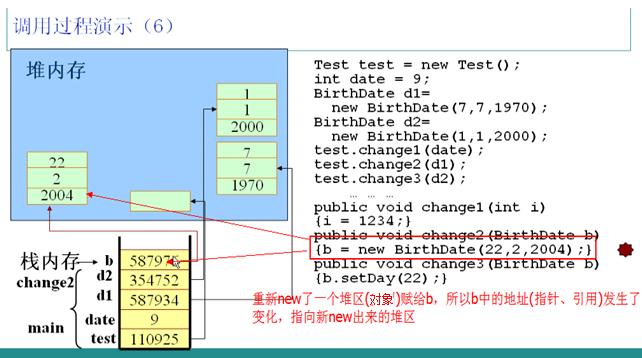
把1234赋给i。很简单的一步。



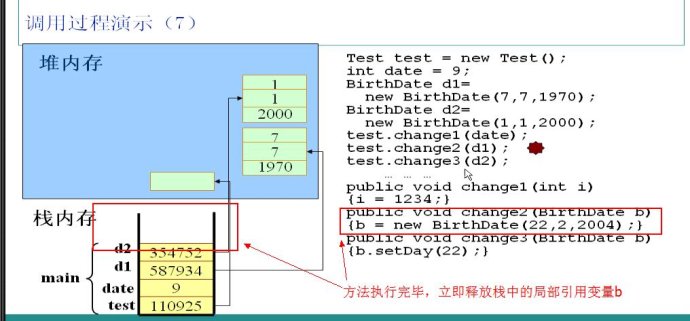
change1方法执行完毕，立即释放局部变量i所占用的栈空间。



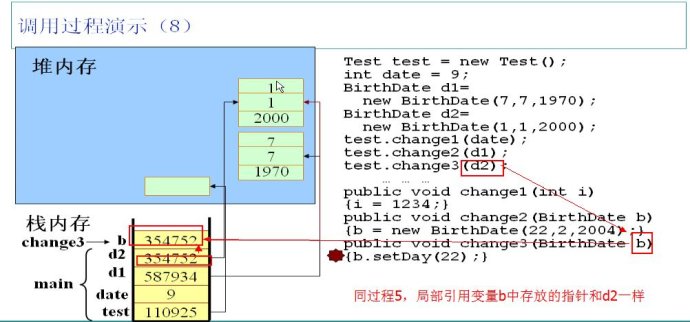
调用test对象的change2方法，以实例d1为参数。JVM检测到change2方法中的b参数为局部变量，立即加入到栈中，由于是引用类型的变量，所以b中保存的是d1中的指针，此时b和d1指向同一个堆中的对象。在b和d1之间传递是指针。



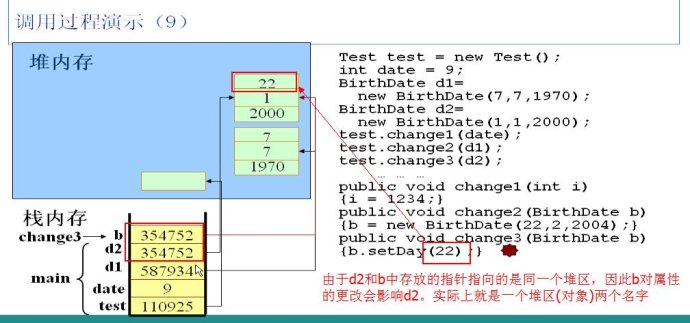
change2方法中又实例化了一个BirthDate对象，并且赋给b。在内部执行过程是：在堆区new了一个对象，并且把该对象的指针保存在栈中的b对应空间，此时实例b不再指向实例d1所指向的对象，但是实例d1所指向的对象并无变化，这样无法对d1造成任何影响。



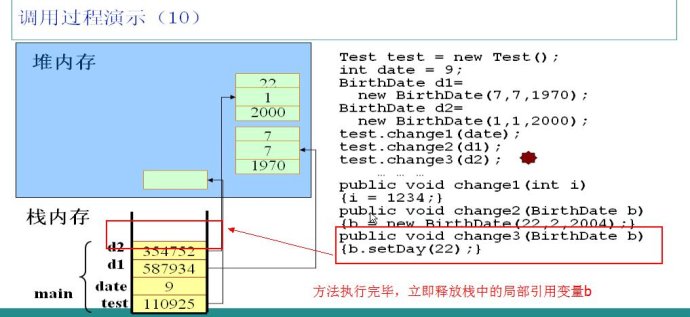
change2方法执行完毕，立即释放局部引用变量b所占的栈空间，注意只是释放了栈空间，堆空间要等待自动回收。



调用test实例的change3方法，以实例d2为参数。同理，JVM会在栈中为局部引用变量b分配空间，并且把d2中的指针存放在b中，此时d2和b指向同一个对象。再调用实例b的setDay方法，其实就是调用d2指向的对象的setDay方法。



调用实例b的setDay方法会影响d2，因为二者指向的是同一个对象。



change3方法执行完毕，立即释放局部引用变量b。

以上就是Java程序运行时内存分配的大致情况。其实也没什么，掌握了思想就很简单了。无非就是两种类型的变量：基本类型和引用类型。二者作为局部变量，都放在栈中，基本类型直接在栈中保存值，引用类型只保存一个指向堆区的指针，真正的对象在堆里。作为参数时基本类型就直接传值，引用类型传指针。

小结：

1.分清什么是实例什么是对象。Class a= new Class();此时a叫实例，而不能说a是对象。实例在栈中，对象在堆中，操作实例实际上是通过实例的指针间接操作对象。多个实例可以指向同一个对象。

2.栈中的数据和堆中的数据销毁并不是同步的。方法一旦结束，栈中的局部变量立即销毁，但是堆中对象不一定销毁。因为可能有其他变量也指向了这个对象，直到栈中没有变量指向堆中的对象时，它才销毁，而且还不是马上销毁，要等垃圾回收扫描时才可以被销毁。

3.以上的栈、堆、代码段、数据段等等都是相对于应用程序而言的。每一个应用程序都对应唯一的一个JVM实例，每一个JVM实例都有自己的内存区域，互不影响。并且这些内存区域是所有线程共享的。这里提到的栈和堆都是整体上的概念，这些堆栈还可以细分。

4.类的成员变量在不同对象中各不相同，都有自己的存储空间(成员变量在堆中的对象中)。而类的方法却是该类的所有对象共享的，只有一套，对象使用方法的时候方法才被压入栈，方法不使用则不占用内存。

以上分析只涉及了栈和堆，还有一个非常重要的内存区域：常量池，这个地方往往出现一些莫名其妙的问题。常量池是干嘛的上边已经说明了，也没必要理解多么深刻，只要记住它维护了一个已加载类的常量就可以了。接下来结合一些例子说明常量池的特性。

预备知识：

基本类型和基本类型的包装类。基本类型有：byte、short、char、int、long、boolean。基本类型的包装类分别是：Byte、Short、Character、Integer、Long、Boolean。注意区分大小写。二者的区别是：基本类型体现在程序中是普通变量，基本类型的包装类是类，体现在程序中是引用变量。因此二者在内存中的存储位置不同：基本类型存储在栈中，而基本类型包装类存储在堆中。上边提到的这些包装类都实现了常量池技术，另外两种浮点数类型的包装类则没有实现。另外，String类型也实现了常量池技术。