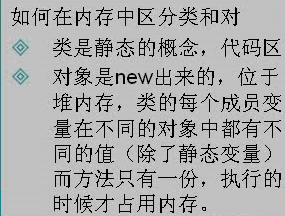
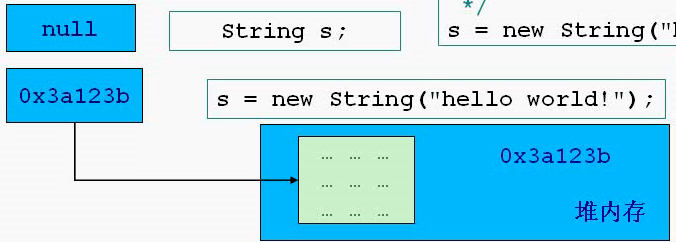
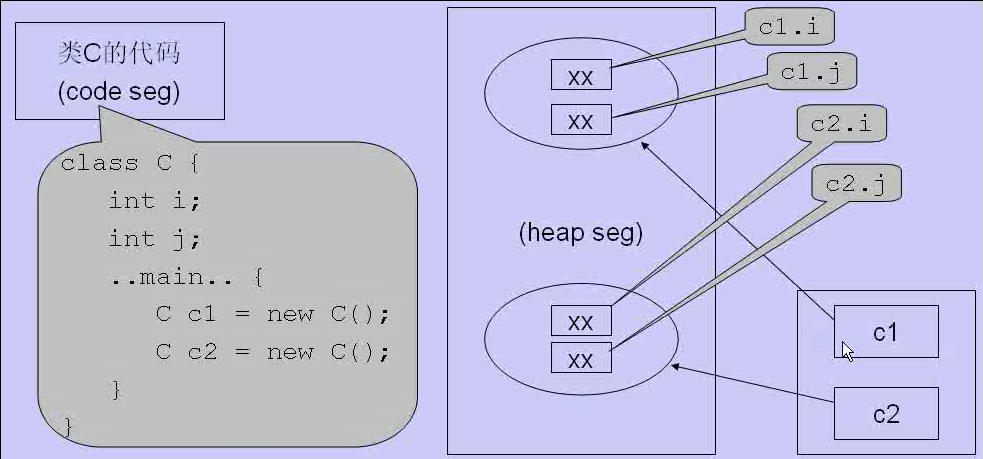
1. Java面向对象基本概念—引用(Java语言中除基本类型之外的变量类型都称之为引用类型，引用类型占两块内存，基本类型占一块内存)，可以理解为C/C++中的指针。

方法只有一份，执行的时候才占用内存。

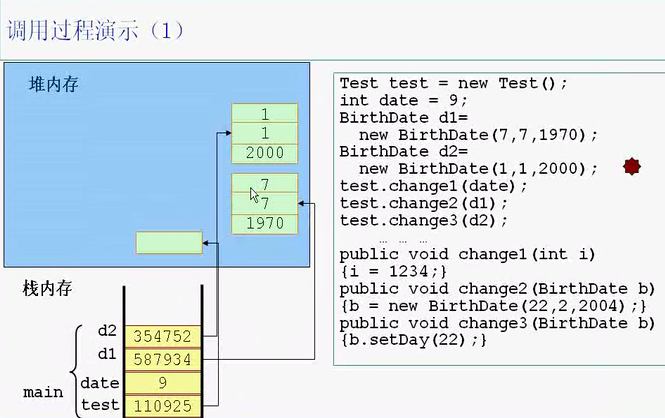


思想：new对象时，要形象的想象到一小块内存(栈)指向一大块内存(堆)。



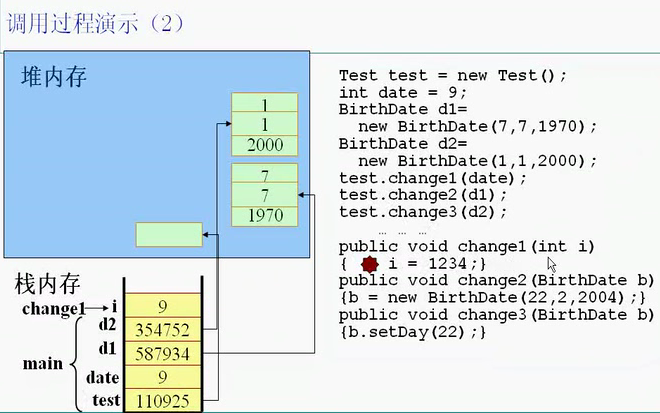
c1、c2为局部变量，在“栈”中分配两个地方来存储，new出来的分配在heap堆中。

1. 实例

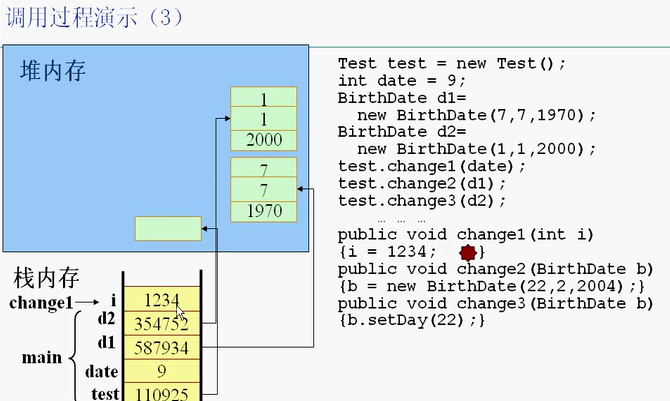


说明(1)：date为基础类型局部变量，分配在栈内存中，并且只占一块空间。

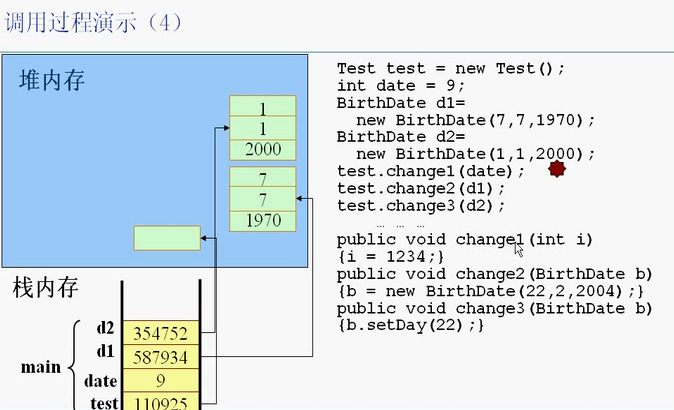
d1为引用类型局部变量，分配在占内存中，指向new出来的存储在堆内存中的内容。



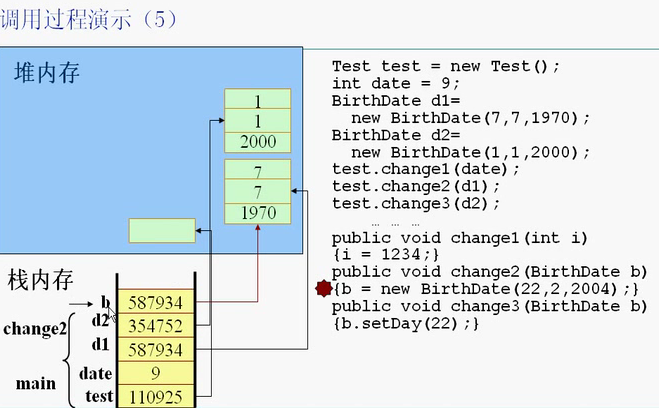
说明(2)：change1方法中的i为局部变量，在栈内存中分配空间，并且采用“值传递”的方法将date的值9传递给i

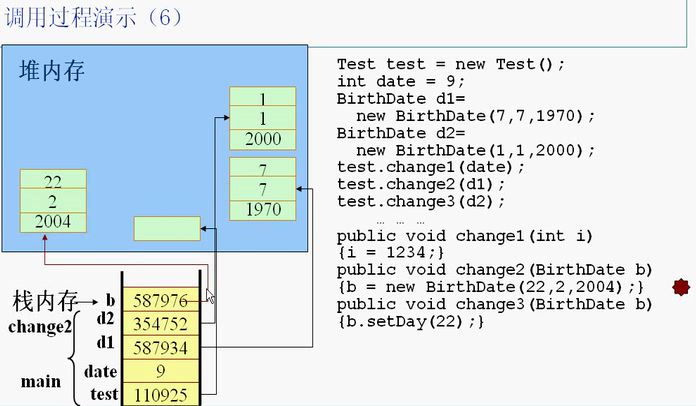


说明(3)：change1方法将局部变量i的值9更改为1234

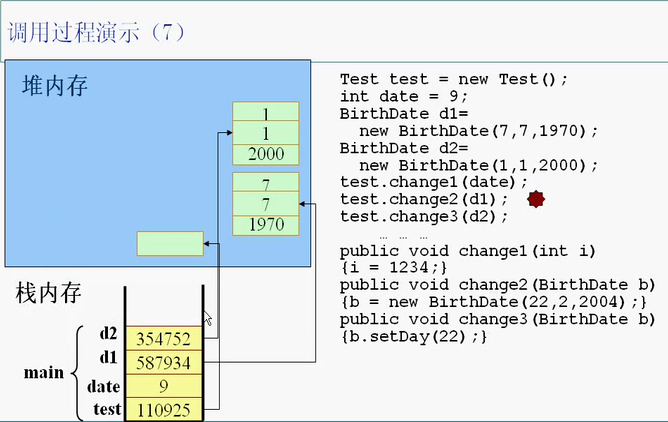


说明(4)：change1方法结束，为该方法非配的所有局部变量的内存空间全部消失，其中的局部变量i就会消失。date的值依然存在，依然是9，所以并没有更改到date的值。

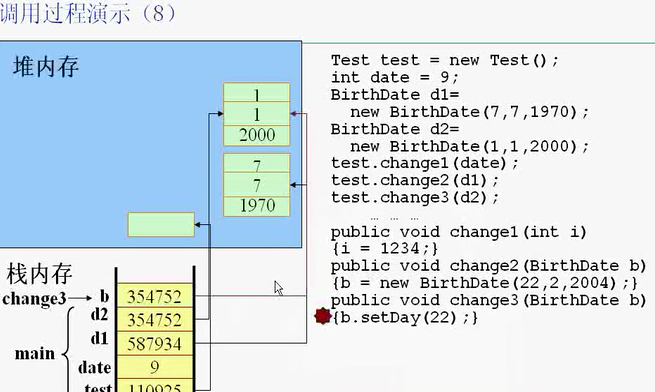


说明(5)：将d1的引用(地址值)传递给change2的局部变量b，这样通过b也可以通过该地址找到同样的对象。即b和d1指向同一对象。  


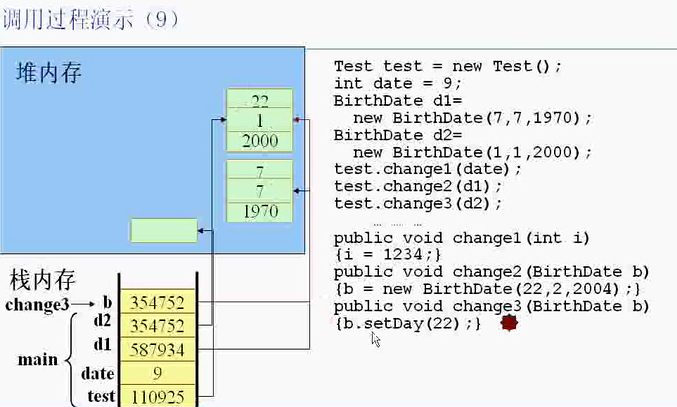
说明(6)：change2方法为b new出来了一个新对象。



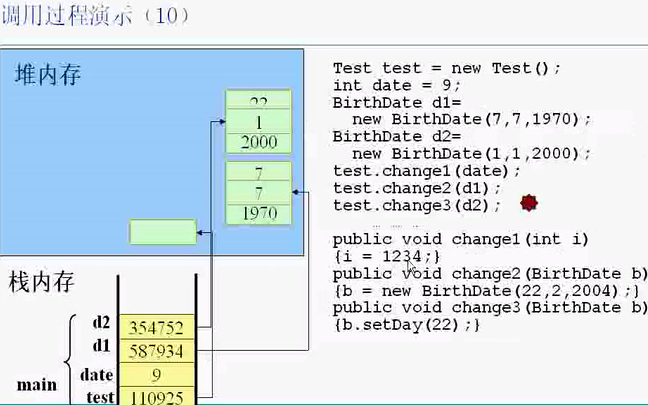
说明(7)：change2方法结束，该方法中的局部变量b也一同消失(对内存中的内容会由垃圾收集器自动回收，时间不定，就认为消失就可以了)。



说明(8)：change3方法开始，将d2的引用传递给局部变量b，此时b与d2指向同一内容。

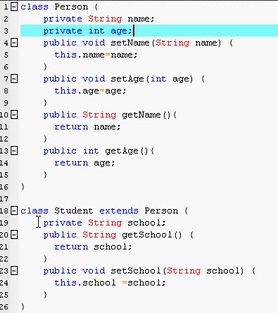


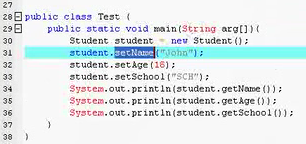
说明(9)：调用b的setDay方法，将其指向的内存区域的值改掉。

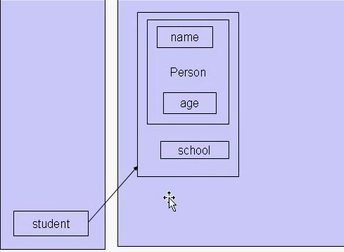


说明(10)：change3方法结束，为其分配的局部变量b消失，但是这时候d2指向的内容被修改了。

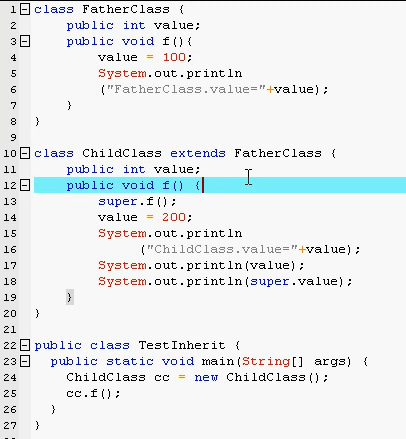
1. 例子：



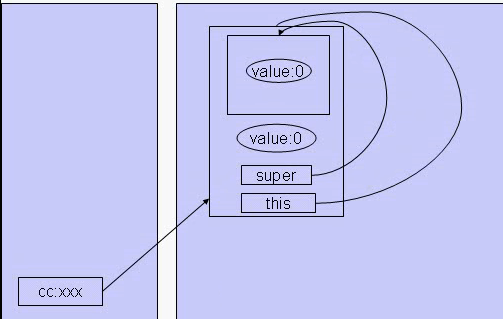




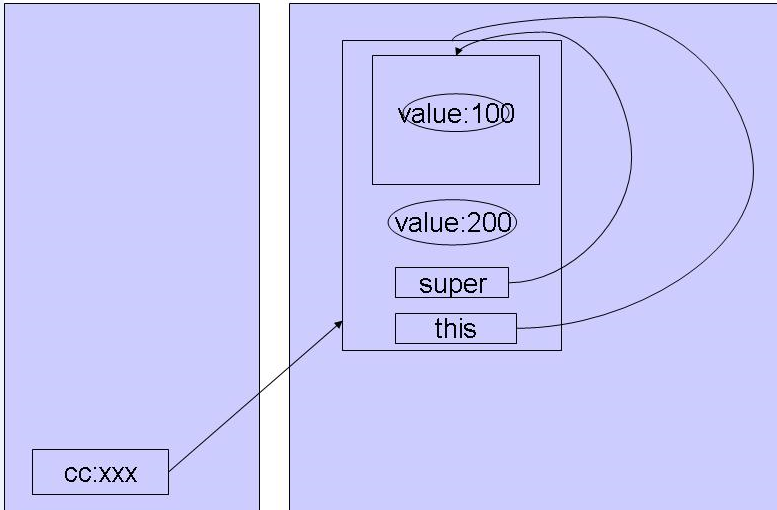
1. 例子2：



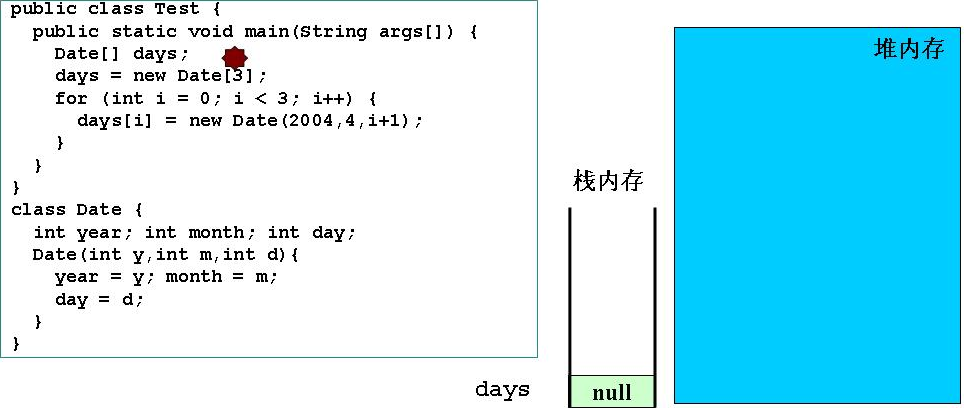
ChildClass cc = new ChildClass()执行完成后内存布局：

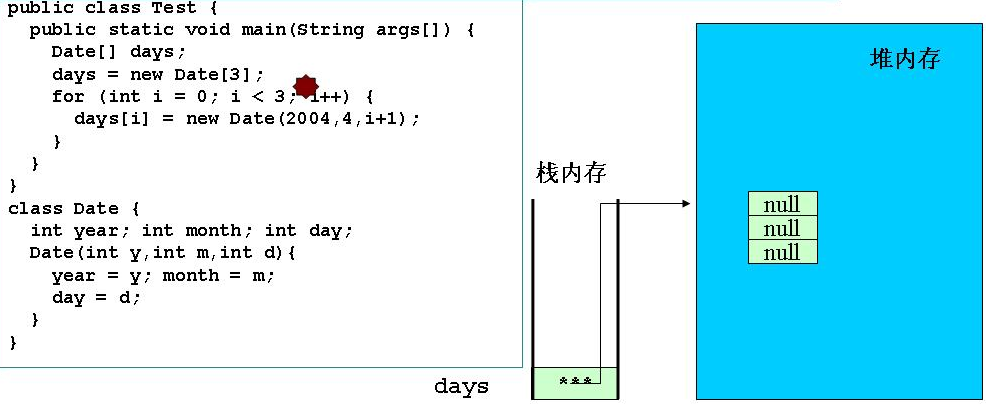


**cc.f()调用完毕后:**

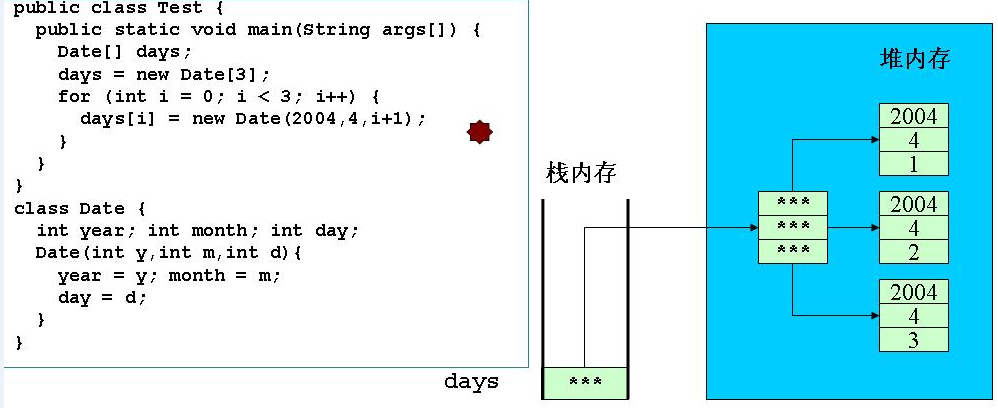


1. 例子3(数组的内存分析)





days中存放的是**引用**。



1. 当**最后一个引用消失时，对象就会变成可回收的**。有三种方法可以释放对象的引用
   1. 引用永久性离开它的范围scope：

void go() {

Life z = new Life();//z会在方法结束时消失

}

* 1. 引用被赋值到其他对象上：

Life z = new Life();

z = new Life();//第一个对象会在z被赋值到别处时挂掉

* 1. 直接将应用设定为null：

Life z = new Life();

z = null;//第一个对象会在z被赋值为null时击毙